Af / IfW

UNITED STATES PATENT AND TRADEMARK OFFICE

JAN 23 2008

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/047,446	01/14/2002	Max Copperman		1061

7590   01/04/2008

MAX COPPERMAN
233 SUNSET AVENUE
SANTA CRUZ, CA 95060

EXAMINER
ABEL JALIL, NEVEEN

ART UNIT	PAPER NUMBER
2165	

MAIL DATE	DELIVERY MODE
01/04/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

PTOL-90A (Rev. 04/07)

	Application No.	Applicant(s)
Office Action Summary	10/047,446	COPPERMAN ET AL.
	Examiner	Art Unit	
	Neveen Abel-Jalil	2165	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 October 2007*.

2a)☒ This action is **FINAL.**   2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3-11,25-31,36-42 and 62-64* is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 3-11, 25-31, 36-42, and 62-64* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☐ All   b)☐ Some * c)☐ None of:

      1.☐ Certified copies of the priority documents have been received.

      2.☐ Certified copies of the priority documents have been received in Application No. _____.

      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *5/7/07*.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____ .

DETAILED ACTION

Remarks

1.   In response to Applicant's Amendment filed on October 3, 2007, claims 1, 3-11, 25-31,

36-42, and 62-64 are pending in the application.

2.   Applicant's response has overcome the previous claim rejections under 112, second.

3.   Applicant's remarks and amendment to the claims changes the scope and focus of the

invention from what appeared to be "preparation" and generation of documents corpus once

extracted and compiled then matched to previously unknown user query (i.e. no knowledge of

what the query will be) to query focused knowledge wherein the extraction and generation is

solely based on the content of the retrieved results to the query. Hence necessitating the new

ground(s) of rejection below.

Information Disclosure Statement

4.   The information disclosure statement (IDS) submitted on May 7, 2007 is in compliance

with the provisions of 37 CFR 1.97. Accordingly, the information disclosure statement is being

considered by the examiner.

Claim Rejections - 35 USC § 102

5.   The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the

basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless --

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

6.   Claims 1, 3-5, 7, 11, 25-28, 31, 36, and 42 are rejected under 35 U.S.C. 102(e) as being

anticipated by Mikheev (U.S. Pub. No. 2002/0055919 A1).

As to claim 1, Mikheev discloses a method of steering an end-user to a document needed

by the end-user, the method including:

receiving from the end-user a user query including language (See page 1, paragraph

0006, wherein it is inherent that a query is formed using query language);

using at least a portion of the user query to search for an retrieve a set of one or more

documents (See Figure 4, wherein keyword search was conducted and results presented

accordingly, also see page 1, paragraph 0006);

extracting from the retrieved set of one or more documents, at least one concept feature

that appears in at least one document in the retrieved set of one or more documents (See Figure

3, 40, 42, and see page 1, paragraph 0007, wherein "concept feature" is read on "phrase" read in

light of Applicant's specification paragraph 0044);

using the at least one concept feature to determine, using a knowledge map, at least one

matched concept that corresponds to the at least one concept feature (See page 2, paragraph

0039, also see Figure 3, 50, wherein "informational map" reads on "knowledge map", wherein

"concept feature" is read on "phrase" read in light of Applicant's specification paragraph 0044);

and

presenting to the end-user at least one indication of the at least one matched concept and

at least one document associated with the at least one matched concept (See Figure 7, shows

concept maps with matched documents, and see page 2, paragraph 0036, wherein "visually

displaying the connected nodes and links on a user interface" reads on "an indication").

As to claim 3, Mikheev discloses further including:

presenting to the user at least one indication of at least one related concept to the at least

one matched concept ((See page 5, paragraph 0062, wherein "displaying the connected nodes

and links on a user interface" reads on "an indication", and wherein "related concept" reads on

"additional concept" linked to the "matched concept");

receiving from the user a selection of at least one related concept (See page 5, paragraph

0062, See Figure 20, all concept selections are linked on the map and displayed to the user); and

presenting to the user at least one indication of at least one document associated with the

user-selected related concept (See page 4, paragraph 0059, wherein "displaying the connected

nodes and links on a user interface" reads on "an indication", and wherein "related concept"

reads on "additional concept" linked to the "matched concept").

As to claims 4, 26, and 27, Mikheev discloses in which the presenting to the user at least

one indication of at least one document associated with the user-selected related concept includes

presenting to the user the at least one indication of the at least one document associated with both

the user-selected related concept and the at least one matched concept (See page 5, paragraph

0062, also see Figure 23, wherein "displaying the connected nodes and links on a user interface"

reads on "an indication", and wherein all concepts are ranked and associated with documents).

As to claim 5, <u>Mikheev</u> discloses further including presenting to the user at least one indication of the at least one matched concept (See corresponding rejection in claim 1 above).

As to claims 7, and 28, <u>Mikheev</u> discloses further including ranking related concepts (See page 5, paragraph 0063, wherein "ranked" reads on "most relevant", wherein if matched concepts themselves are ranked, it is inherent that any other concepts can be ranked too, wherein "related concepts" are read on "additional concepts" from the refined search).

As to claim 11, <u>Mikheev</u> discloses a computer-readable medium for performing the method of claim 1 (See claim 1 rejection above).

As to claim 25, <u>Mikheev</u> discloses a method of steering an end-user to a document needed by the end-user, the method including:

receiving from the end-user a user query including language (See corresponding rejection in claim 1 above);

searching for and retrieving a set of one or more documents by determining whether at least one feature in the user query language substantially matches at least concept feature associated with at least one concept in a plurality of concepts in a knowledge map that are pregrouped into a plurality of groups, each concept including as evidence at least one concept feature (See page 2, paragraph 0039, also see Figure 3, 50, wherein "informational map" reads

on "knowledge map", wherein "concept feature" is read on "phrase" read in light of Applicant's

specification paragraph 0044, wherein "pregrouped" reads on "clusters");

extracting from the retrieved set of one or more documents, at least one concept feature

that appears in at least one document in the retrieved set of one or more documents (See

corresponding rejection in claim 1 above);

using the at least one concept feature to determine at least one matched concept that

corresponds to the at least one concept feature (See corresponding rejection in claim 1 above);

presenting to the end-user, when the at least one feature in the user query language

substantially matches the at least one concept feature associated with a concept, wherein the at

least one concept feature is obtained from the set of one or more documents, at least one

indication of the at least one matched concept and at least one related concept to the at least one

matched concept , the at least one related concept determined from a predefined correspondence

relationship between the at least one matched concept and the at least one related concept, the

indication of the at least one related concept presented as corresponding to the at least one

matched concept to which it is related (See page 2, paragraph 0037, wherein "predefined

relationship" reads on "similar subject matter grouped together", and see page 5, paragraph

0066); and

presenting to the end-user, when the at least one feature in the user query language

substantially matches the at least one concept feature associated with the at least one concept, at

least one indication of the at least one matched concept and at least one document associated

with the at least one matched concept, the at least one document drawn from a plurality of

documents that are respectively linked to one or more of the concepts in the knowledge map (See

Figure 22, also see page 5, paragraph 0062, wherein "concept feature" is read on "phrase" read

in light of Applicant's specification paragraph 0044).

As to claim 31, <u>Mikheev</u> discloses a computer-readable medium for performing the

method of claim 25 (See claim 25 rejection).

As to claim 36, <u>Mikheev</u> discloses a method of steering a user to a document needed by

the end-user, the method including:

receiving from the end-user a user query including language (See corresponding rejection

in claim 1 above);

searching and retrieving a set of one or more documents determining whether at least one

feature in the user query language substantially matches at least one concept feature associated

with a concept in a plurality of concepts in a knowledge map that are pre-grouped into a plurality

of primary groups, each concept including as evidence at least one concept feature that is also in

at least one document in a plurality of documents that are tagged to one or more of the concepts

in the knowledge map, wherein the at least one concept feature is extracted from the retrieved set

of one or more documents (See corresponding rejection in claim 25 above);

presenting to the end-user, when the at least one feature in the user query language

substantially matches the at least one concept feature associated with the concept (See rejections

in claim 1, and 25 above):

at least one indication of the at least one matched concept (See corresponding rejection in

claim 2 above);

at least one indication of at least one related concept to the at least one matched concept

(See corresponding rejection in claim 2 above); and

at least one indication of at least one document associated with the at least one matched

concept (See corresponding rejection in claim 2 above).

As to claim 42, <u>Mikheev</u> discloses a computer-readable medium for performing the

method of claim 36 (See claim 36 rejection above).

Claim Rejections - 35 USC § 103

7.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8.   Claims 6, 8, 9-10, 29-30, and 62-64 are rejected under 35 U.S.C. 103(a) as being

unpatentable over <u>Mikheev</u> (U.S. Pub. No. 2002/0055919 A1) in view of <u>Korda et al.</u> (U.S.

Patent No. 6,564,210 B1).

As to claim 6, <u>Mikheev</u> discloses the claimed invention and teaches relevance of concepts

to query results but does not explicitly teach in which the presenting to the user at least one

indication of the at least one matched concept and the presenting to the user at least one related

concept to the at least one matched concept includes presenting to the user a paired indication of:

(1) a matched concept, and (2) a corresponding related concept.

Korda et al. teaches in which the presenting to the user at least one indication of the at least one matched concept and the presenting to the user at least one related concept to the at least one matched concept includes presenting to the user a paired indication of: (1) a matched concept, and (2) a corresponding related concept (See column 7, lines 35-67, and see column 8, lines 1-27).

It would have been obvious to one having ordinary skill in the art at the time the invention was made to modify the teachings of Mikheev with the teachings of Korda et al. to include presenting to the user at least one indication of the at least one matched concept and the presenting to the user at least one related concept to the at least one matched concept includes presenting to the user a paired indication of: (1) a matched concept, and (2) a corresponding related concept because it provides for more customized and easier presentation of search results.

As to claims 8, and 62, Mikheev does not explicitly teach in which the presenting to the end-user at least one indication of at least one related concept to the at least one matched concept includes presenting to the end-user ranked indications of related concepts .

Korda et al. teaches in which the presenting to the end-user at least one indication of at least one related concept to the at least one matched concept includes presenting to the end-user ranked indications of related concepts (See rejection for claim 6 above).

The motivation to combine is similar to claim 6 above.

As to claims 9, 29, and 63, <u>Mikheev</u> teaches the claimed invention but does not explicitly teaches in which the ranking related concepts includes ranking using a number of times that the related concept was previously-selected by at least one end-user.

<u>Korda et al.</u> teaches in which the ranking related concepts includes ranking using a number of times that the related concept was previously-selected by at least one end-user (See column 9, lines 5-25, wherein "repeated searches" reads on "preciously selected", and see column 10, lines 40-57).

It would have been obvious to one having ordinary skill in the art at the time the invention was made to modify the teachings of <u>Mikheev</u> with the teachings of <u>Korda et al.</u> to include ranking related concepts includes ranking using a number of times that the related concept was previously-selected by at least one end-user because it provides for improved ranking and better accuracy of search result retrieval.

As to claims 10, 30, and 64, <u>Mikheev</u> as modified teaches including promoting a related concept in the ranking when a previous selection by an end-user resulted in an inferred success in returning at least one relevant document (See <u>Korda et al.</u> column 9, lines 5-25, wherein "implicit" reads on "inferred").

Claim Rejections - 35 USC § 103

9.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person

having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

10.   Claims 37-41 are rejected under 35 U.S.C. 103(a) as being obvious over Mikheev (U.S.

Pub. No. 2002/0055919 A1).

As to claim 37, Mikheev discloses the claimed invention but does not explicitly recite in

which the related concept is obtained from a derived group mapping relationships between

primary group concept nodes from the same or different primary groups. Mikheev teaches

clusters, sub clusters, and related clusters with additional mapping.

It would have been obvious to one having ordinary skill in the art at the time the

invention was made to assign different headings to primary concept groups since it is known in

the database art that concept groups are user definable (non-functional descriptive material does

not add functionality to the claim and any type of content can be stored and defined in a

knowledge bases) (see In re Gulack, 703 F.2d 1381, 1385, 217 USPQ 401, 404 (Fed. Cir. 1983);

In re Lowry, 32 F.3d 1579, 32 USPQ2d 1031 (Fed. Cir. 1994).).

As to claim 38, Mikheev as modified teaches in which the primary groups or derived

groups including an Activities group, a Symptoms group, a Products group, and an Objects

group (See corresponding rejection for claim 37 above) Specific to the claim language of:

further including obtaining a related concept to the at least one matched concept from a

derived group that includes **at least one of**:

an Activities and Objects group, including at least one relationship between an Activities

concept and an Objects concept;

an Activities and Products group, including at least one relationship between an

Activities concept and a Products concept;

a Symptoms and Objects group, including at least one relationship between a Symptoms

concept and an Objects concept;

a Symptoms and Products group, including at least one relationship between a Symptoms

concept and a Products concept; and

a Symptoms and Activities group, including at least one relationship between a

Symptoms concept and an Activities concept.

As to claim 39, is rejected under the same rational as claim 38 wherein Mikheev as

modified teaches:

further including obtaining a related concept to the at least one matched concept from a

derived group that includes **at least one of**:

an Activities and Activities group, including at least one relationship between different

Activities concepts;

an Objects and Objects group, including at least one relationship between different

Objects concepts;

a Symptoms and Symptoms group, including at least one relationship between different

Symptoms concepts; and

a Products and Products group, including at least one relationship between different

Products concepts.

As to claim 40, <u>Mikheev</u> as modified discloses further including obtaining a related

concept to the at least one matched concept from a derived group that includes **at least one of**:

at least one lexically-similar group, including at least one relationship between lexically

similar concepts; and

at least one semantically-similar group, including at least one relationship between

semantically similar concepts.

As to claim 41, is rejected under the same rational as claim 36 wherein <u>Mikheev</u> as

modified teaches the primary groups consist only of Products, Activities, Symptoms, and Objects

groups.

The motivation to include such teachings is the same as motivation for claim 36.

Response to Arguments

11.   Applicant's arguments with respect to claims 1, 3-11, 25-31, 36-42, and 62-64 have been

considered but are moot in view of the new ground(s) of rejection.

Conclusion

12.   Applicant's amendment necessitated the new ground(s) of rejection presented in this

Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a).

Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE

MONTHS from the mailing date of this action. In the event a first reply is filed within TWO

MONTHS of the mailing date of this final action and the advisory action is not mailed until after

the end of the THREE-MONTH shortened statutory period, then the shortened statutory period

will expire on the date the advisory action is mailed, and any extension fee pursuant to 37

CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event,

however, will the statutory period for reply expire later than SIX MONTHS from the date of this

final action.

13.   The prior art made of record and not relied upon is considered pertinent to applicant's

disclosure.

    Wilcox et al. (U.S. Pub. No. 2002/0049792 A1) teaches conceptual content delivery.

    Kato et al. (U.S Pub. No. 2002/0120451 A1) teaches extracting features from retrieved

results.

    Szabo (U.S. Patent No. 6,868,525 B1) teaches visualization of concept hierarchy.

    Alonso et al. (U.S. Patent No. 7,092,936 B1) teaches search and recommendation based

on usage.

    Paik et al. (U.S. Patent No. 6,263,335 B1) teaches information extraction using concept

relationships.

    Nomoto et al. (EP 0822503 A1) teaches linguistic features extracted from the query are

classified into Concepts expressing content of the query.

    Gallivan et al. (U.S. Patent No. 6,978,274 B1) teaches extracting features from

unstructured document and normalizing them into concepts.

Fratkina et al. (U.S Pub. No. 2005/0055321 A1) teaches mutlti-step dialog with user and assigning concept classification.

Perro et al. (U.S. Pub. No. 2002/0152202 A1) teaches extracting keywords from query results.

Adler et al. (U.S. Pub. No. 2003/0033295 A1) teaches filter query results and build thesaurus.

For complete list of cited relevant prior art, see PTO-Form 892.

14.   Any inquiry concerning this communication or earlier communications from the examiner should be directed to Neveen Abel-Jalil whose telephone number is 571-272-4074. The examiner can normally be reached on 8:30AM-5:30PM EST.
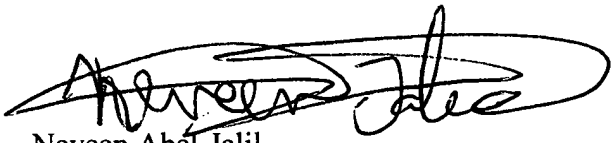
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Christian Chace can be reached on 571-272-4190. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

Neveen Abel-Jalil
Primary Examiner
January 1, 2007

Substitute for form 1449A/PTO **INFORMATION DISCLOSURE STATEMENT BY APPLICANT** *(Use as many sheets as necessary)*	Complete if Known	
	Application Number	10/047,446
	Filing Date	January 14, 2002
	First Named Inventor	Copperman, Max
	Group Art Unit	2165
	Examiner Name	Abel-Jalil, Neveen
Sheet 1 of 1	Attorney Docket No: 1546.015US1	

US PATENT DOCUMENTS

Examiner Initial *	USP Document Number	Publication Date	Name of Patentee or Applicant of cited Document	Filing Date If Appropriate
NPD	US-20010037328A1	11/01/2001	Pustejovsky, J. D., et al.	12/19/2000
	US-20020052894A1	05/02/2002	Bourdoncle, F. , et al.	08/14/2001
	US-20020103798A1	08/01/2002	Abrol, M. S., et al.	02/01/2001
	US-20020169771A1	11/14/2002	Kenneth, L. M., et al.	05/09/2001
	US-20060140030A1	06/29/2006	Bedarida, L. B., et al.	03/24/2005
	US-5,508,959	04/16/1996	Lee, R. R., et al.	03/28/1995
	US-5,694,559	12/02/1997	Samuel, D. H., et al.	03/07/1995
	US-6,034,882	03/07/2000	Johnson, Mark G., et al.	11/16/1998
	US-6,401,084	06/04/2002	Ortega, Ruben E., et al.	03/02/2000
	US-6,411,950	06/25/2002	Michael, Z. M., et al.	11/30/1998
	US-6,515,344	02/04/2003	Wollesen, D. L.	10/30/2000
	US-6,574,622	06/03/2003	Miyauchi, T. K., et al.	08/27/1999
	US-6,686,791	02/03/2004	Zheng, B. C., et al.	04/25/2002
	US-6,920,448	07/19/2005	Kincaid, R. H., et al.	12/19/2001
	US-7,089,236	08/08/2006	Jeffrey, M. S.	10/13/1999
V	US-7,102,951	09/05/2006	Paillet, F. K., et al.	11/01/2004

EXAMINER		DATE CONSIDERED	10/10/07

		Application/Control No.	Applicant(s)/Patent Under Reexamination
Notice of References Cited		10/047,446	COPPERMAN ET AL.
		Examiner	Art Unit	Page 1 of 1
		Neveen Abel-Jalil	2165	

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-6,263,335	07-2001	Paik et al.	707/5
*	B	US-2002/0049792	04-2002	Wilcox et al.	707/522
*	C	US-2002/0055919	05-2002	Mikheev, Andrei	707/3
*	D	US-2002/0120451	08-2002	Kato et al.	704/258
*	E	US-6,457,004	09-2002	Nishioka et al.	707/5
*	F	US-2002/0152202	10-2002	Perro et al.	707/3
*	G	US-2003/0033295	02-2003	Adler et al.	707/3
*	H	US-6,564,210	05-2003	Korda et al.	707/3
*	I	US-2004/0024739	02-2004	Copperman et al.	707/1
*	J	US-6,868,525	03-2005	Szabo, Andrew	715/738
*	K	US-2005/0055321	03-2005	Fratkina et al.	706/045
*	L	US-6,978,274	12-2005	Gallivan et al.	707/102
*	M	US-7,092,936	08-2006	Alonso et al.	707/3

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N	EP 0822503 A1	02-1998	Japan	Nomoto, Masako	G06F 17/30
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	H. L. Wang et al. Semantic Search on Internet Tablular Information Extraction for Answering Queries. CIKM 2000. ACM 2000.
	V	Carl Gutwin et al. Improving browsing in digital liberaries with keyphrase indexes. Decision Support Systems 27 (nov. 1999). Pages 81-104.
	W	Zhixiang Chen et al. Real-time adaptive feature and document learning for web searchVolume 52, Issue 8 , Pages 655 - 665. Published Online: 27 Apr 2001.
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

(54) **Document retrieval system**

(57) Linguistic features extracted from the query are classified into concepts expressing content of the query, and linguistic features extracted from the document are classified into concepts expressing content of the document. The user confirms of which concept the input statement and the document are comprised and which kind of linguistic feature corresponds to each concept, and the system assists the user to adequately select a document, which is closer to the intention of retrieval.
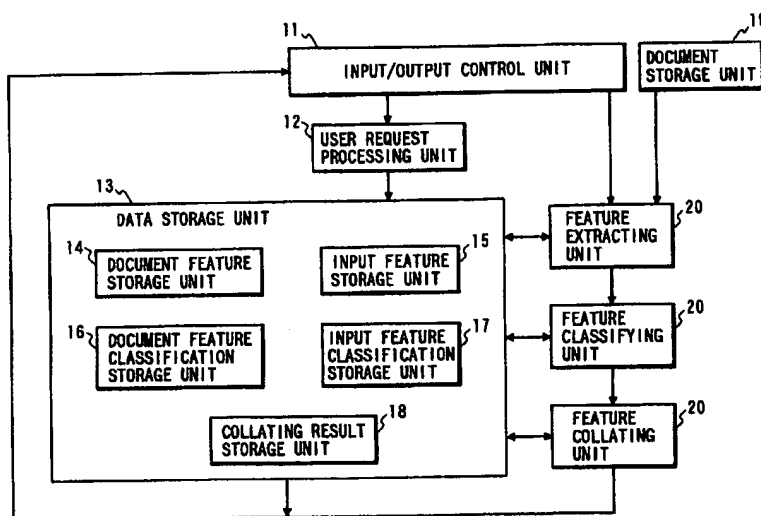
FIG. 1

Description

BACKGROUND OF THE INVENTION

5 [Field of the Invention]

The present invention relates to a document retrieval system for retrieving a document suitable for a user's intention of retrieval from electronic documents.

10 [Description of the Related Art]

The following two retrieval systems have been proposed in the past: a retrieval system based on exact match technique, in which a user inputs queries comprising character strings and logic operators and obtains a document assembly, which satisfy the condition, and a retrieval system based on partial match technique, in which the similarity between
15 the input by the user and the document to be retrieved are compared and collated with each other by some measure, and the documents are ranked according to the closeness to the user's intention of retrieval.

The retrieval based on the exact match technique is advantageous in that the query inputted by the user clearly corresponds to the document assembly as the result of retrieval. However, the vocabulary used in the document to be retrieved is often unclear to the user, and it is difficult to specify a suitable keyword in the query, and in case a large
20 amount of document assemblies have been obtained as the result of retrieval it is so hard for a user to choose relative documents one by one.

On the other hand, the retrieval based on the partial match technique is advantageous in that ranking is assigned to the documents as the result of retrieval according to the closeness to the user's intention of retrieval, while it is not necessarily clear to the user as to what kind of document are ranked on what basis.

25

SUMMARY OF THE INVENTION

In the present invention, linguistic features expressing content of query and document are classified into concept, which expresses each content, to clarify the query and the document, and the concepts which comprise linguistic fea-
30 ture is shown of the document to be retrieved actually corresponding to each concept, which comprises the query and document, and to support in efficiently selecting the document, which is closer to the user's intention of the retrieval.

To solve the above problems, the document retrieval system according to the present invention comprises an input/output control unit for receiving input from a user and for showing result of processing to the user, a user request processing unit for receiving a request other than query among the inputs from the user and for processing content of
35 the request, a document storage unit for storing a document to be retrieved, a feature extracting unit for extracting linguistic feature of the query inputted from said input/output control unit as input feature or for extracting linguistic feature of the document stored in the document storage unit as document feature, an input feature storage unit for storing input feature extracted from the by the feature extracting unit, a document feature storage unit for storing linguistic feature taken out from the document by the feature extracting unit, a feature classifying unit for classifying the input feature
40 stored in the input feature storage unit to correspond to concept of the content expressed by the query or for classifying the document feature stored in the document feature storage unit to correspond to concept of the content expressed by the document, an input feature classification storage unit for storing correspondence between the input feature classification and the input feature as the result of classification of the input feature by the feature classifying unit, and a document feature classifying storage unit for storing correspondence between the document feature classification and
45 document feature as a result of classification of document feature by the feature classifying unit, whereby, in response to a request from the user via the user request processing unit, the content of the query or the specified document is presented to the user via the input/output control unit as corresponding relationship between group of concepts expressed by the input feature classification and the document feature classification and the linguistic feature belonging to each concept group.

50 According to the present invention, the user can confirm easily the user which concept the query or the document is comprised of, and further which linguistic feature corresponds to each concept, and hence, efficiently select the document, which is closer to the intention of retrieval.

The system according to Claim 1 of the present invention comprises an input/output control unit for receiving input from a user and for presenting result of processing to the user, a user request processing unit for receiving a request
55 other than query among the inputs from the user and for processing content of the request, a document storage unit for storing a document to be retrieved, a feature extracting unit for extracting linguistic feature of the query inputted from said input/output control unit as input feature or for extracting linguistic feature of the document stored in the document storage unit as document feature, an input feature storage unit for storing input feature extracted from the query by the

feature extracting unit, a document feature storage unit for storing linguistic feature taken out from the document by the feature extracting unit, a feature classifying unit for classifying the input feature stored in the input feature storage unit to correspond to partial concept of the content expressed by the input statement or for classifying the document feature stored in the document feature storage unit to correspond to concept of the content expressed by the document, an input feature classification storage unit for storing correspondence between the input feature classification and the input feature as the result of classification of the input feature by the feature classifying unit, and a document feature classifying storage unit for storing correspondence between the document feature classification and document feature as a result of classification of document feature by the feature classifying unit, and, in response to the request sent from the user via the user request processing unit, contents of the query or the specified document are presented to the user via the input/output control unit as corresponding relationship between the group of concepts expressed by the input feature classification and the document feature classification and the linguistic feature belonging to each concept group, and correspondence between the concept expressing the content of the query by the input feature classification and the input feature belonging to each input feature classification as well as correspondence between the concept expressed by the document feature classification of the document and the document feature belonging to each document feature classification are presented in response to the request of the user, the system thus helps the user in easily recognizing which concept the input statement or the document is comprised of, and further, which kind of linguistic feature corresponds to each concept.

In the system according to Claim 2 of the present invention, in response to the request from the user, the input feature classification stored in the input feature classification storage unit and the correspondence between the input feature classification and the input feature of the specified document and the document feature classification stored in the document feature classification storage unit and the correspondence between the document feature classification and the document feature, and further, the document feature stored in the document feature storage unit are added, deleted or corrected, whereby the contents of the input statement and the document are expressed more adequately.

The system according to Claim 3 of the present invention comprises a feature collating unit for collating an input feature classification stored in the input feature classification storage unit and the corresponding input feature with the document feature classification stored in the document feature classification storage unit and the corresponding document feature, and a collating result storage unit for storing collating method and result by the feature collating unit, whereby the contents of the query and the document are compared and collated with the concept expressed by the input feature classification and the input feature belonging to the concept and the concept expressed by the document feature classification and the document feature belonging to the concept, and, as the result of calculation of similarity between the query and the document, not only the score and the ranking of each of the documents finally obtained but also method and result of collating of the similarity are presented to the user, thereby demonstrating to the user at which viewpoint the collating has been performed and how each document has been evaluated.

In the system according to Claim 4 of the present invention, when the query is collated with the accumulated document, a specific document feature classification and the document feature selected by the user via the user request processing unit are treated as provisional input feature classification or input feature, the feature collating unit collates them with the document feature classification and the document feature of each document, and the result of collating is presented to the user, whereby, by regarding the specific document feature classification and the document feature provisionally as the input feature classification and the input feature and collecting them with the document, if there is any concept or linguistic feature not appearing in the original input feature classification or the input feature and being regarded as adequate as the input feature classification or the input feature, these are collated with the document and the effect can be easily confirmed by the user.

In the system according to Claim 5 of the present invention, when the query is collated with the accumulated document, weight is set to the input feature classification or the input feature by the degree of importance specified by the user via the user request processing unit, the feature collating unit collates the input feature classification or the input feature thus weighted with the document feature classification or the document feature of each document, and the result of collating is presented to the user, whereby the user gives the degree of importance to the input feature classification or the input feature as a concept to constitute the input statement or as linguistic feature, to clearly demonstrate the intention of retrieval, and accuracy of retrieval is increased.

BRIEF DESCRIPTION OF THE DRAWINGS

The object and features of the present invention will become more readily apparent from the following detailed description of the preferred embodiments taken in conjunction with the accompanying drawings in which:

Fig. 1 is a block diagram showing functional arrangement of a document retrieval system in Embodiment 1 of the present invention;
Fig. 2 shows an example of query in Embodiment 1 of the present invention;

Fig. 3 gives examples of data of an input feature storage unit in Embodiment 1 of the present invention;

Fig. 4 represents an example of hierarchical thesaurus in Embodiment 1 of the present invention;

Fig. 5 shows a first example of data of an input feature classification storage unit in Embodiment 1 of the present invention;

Fig. 6 shows examples of data of a document feature classification storage unit in Embodiment 1 of the present invention;

Fig. 7 shows a first example of a document ranking table in Embodiment 1 of the present invention;

Fig. 8 shows a second example of the data of the input feature classification storage unit in Embodiment 1 of the present invention;

Fig. 9 shows a second example of a document ranking table in Embodiment 1 of the present invention;

Fig. 10 shows an example of provisional input feature classification and input features in Embodiment 1 of the present invention;

Fig. 11 shows a third example of a document ranking table in Embodiment 1 of the present invention;

Fig. 12 shows a third example of data of the input feature classification storage unit in Embodiment 1 of the present invention;

Fig. 13 gives a fourth example of data of the input feature classification storage unit in Embodiment 1 of the present invention; and

Fig. 14 shows a fourth example of the document ranking table in Embodiment 1 of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following, description will be given on embodiments of the present invention referring to Fig. 1 to Fig. 14.

Fig. 1 is a block diagram showing functional arrangement of a document retrieval system according to an embodiment of the present invention. In Fig. 1, reference numeral 11 represents an input/output control unit, 12 represents a user request processing unit, 13 is a data storage unit, 14 is a document feature storage unit, 15 is an input feature storage unit, 16 is a document feature classification storage unit, 17 is an input feature classification storage unit, 18 is a collating result storage unit, 19 is a document storage unit, 20 is a feature classifying unit, and 22 is a feature collating unit.

In the following, description will be given on operation of the document retrieval system with the above arrangement. First, a query described in natural language is inputted from a user via the input/output control unit 11. The feature extracting unit 20 analyzes the input statement, extracts important words and phrases as linguistic features, and if necessary, these are stored in the input feature storage unit 15 together with statistical information such as frequency of appearance of these features and degree of importance . It is also possible to describe these input features by developing them to homonyms, synonyms, narrower term, etc. using a thesaurus.

Fig. 2 shows examples of a query. Fig. 3 summarizes examples of data stored in the input feature storage unit 15 in case unnecessary words such as symbols, particles, etc. are exempted from the words obtained through morphological analysis to the query of Fig. 2 and the remaining words are regarded as input features.

On the other hand, the feature extracting unit 20 analyzes each of the documents stored in the document storage unit 19 in similar manner. Important words and phrases are extracted as linguistic features, and if necessary, these are stored in the document feature storage unit 14 together with statistical information such as frequency of appearance of these features and degree of importance.

When the feature extracting unit 20 extracts important words and phrases, for example, information such as frequency of the words, distribution of words among documents, parts of speech, appearing position in the document, syntactic and semantic relationship with other words, etc. are used for the judgment of the degree of importance of the words.

Next, the feature classifying unit 21 classifies the input features stored in the input feature storage unit 15 to correspond to each concept, which comprises the query, puts a classification name and stores it to the input feature storage unit 17. To classify the input features, it is supposed that the feature classifying unit 21 possesses hierarchical thesauruses. Nodes at a given depth on the hierarchical thesauruses are set as criteria for classification, and the words having semantically closer concept under the node are put together. Also, there is a method to put together specific words having closer syntactic or semantic relationship using concurrence dictionary or concept dictionary. Fig. 4 gives an example of a part of the hierarchical thesaurus possessed by the feature classifying unit 21.

As the classification name to express coherence or grouping of the input features, a word with a concept corresponding to the node at a given depth in hierarchical thesaurus possessed by the feature classifying unit 21 with respect to the input feature of a certain group and with broader concept of a plurality of words is used. In addition, there are a method to use one of homonyms or synonyms in the thesaurus or a method to use one of the words belonging to the same input feature classification. Fig. 5 shows examples of data stored in the input feature classification storage unit 17 such as input feature corresponding to the input feature classification obtained using the broader concept 1 as criterion

for classification in the hierarchical thesaurus of Fig. 4.

Further, the feature classifying unit 21 also classifies the document feature stored in the document feature storage unit 14 so that it corresponds to concept comprising the document, puts a classification name and stores it in the document feature classification storage unit 16. The method to classify the document feature and to determine the classification name is the same as in the classification of the above input feature. Fig. 6 shows examples of data stored in the document feature classification storage unit 16 such as correspondence between the document feature classification and the document feature. In Fig. 6, the document feature classification common to the input feature classification and the document feature common to the input feature are enclosed by the symbol "[]", and identifiers such as "A", "B", etc. are put to the input feature classification and the document feature classification. These notations and symbols will be also used in the description hereinafter.

Next, the feature collating unit 22 collates the document feature classification and the corresponding document feature stored in the document feature classification storage unit 16 with the input feature classification and the corresponding input feature stored in the input feature classification storage unit 17 and determines the ranking of the documents. The results of the collating of the document and a document ranking table indicating the ranking are stored in the collating result storage unit 18, and the document ranking table is presented to the user via the input/output control unit 11.

Description will be given on an example of collating of the document feature classification and the document feature with the input feature classification and the input feature, referring to the examples of the data in the input feature classification storage unit 17 shown in Fig. 5 and to the examples of the data of the document feature classification storage unit 16 given in Fig. 6.

As an example of a method to calculate score of each document, evaluation function is used as follows:

$$\text{Score } E(\alpha) \text{ of the document } \alpha = \Sigma \text{ (Weight of document feature classification to which document feature belongs } \times \text{ Weight of document feature } \times \text{ Frequency of appearance of the document feature in the document } \alpha)$$

Equation 1:

In case the input feature classification and the input feature are not weighted, the document feature classification and the document feature are evaluated depending upon whether the corresponding input feature classification and input feature are present or not. As the method to determine the weight of each document feature classification and the document feature, it is supposed that weight is given according to:

Rule 1:

(1a) to give weight 1 to the document feature classification having the corresponding input feature classification;
(1b) to give weight 1 to the document feature of each document feature classification having the corresponding input feature; and
(1c) to give weight 0 to all of the document feature classification and document feature other than (1a) and (1b) above. Because the in-document frequency of appearance of the document feature of Fig. 6 is 1 in all cases, the scores of the documents are obtained by the equation 1 as follows:

Document 1:

$$E(1) = 1 \times 1 \times 1 + 1 \times 1 \times 1 + 1 \times 1 \times 1 + 1 \times 1 \times 1 +$$
$$0 \times 0 \times 1 + 0 \times 0 \times 1 + 0 \times 0 \times 1 + 0 \times 0 \times 1 +$$
$$0 \times 0 \times 1 + 0 \times 0 \times 1 + 0 \times 0 \times 1$$
$$= 4$$

Document 2:

$$E(2) = 1 \times 1 \times 1 + 0 \times 1 \times 1 + 0 \times 0 \times 1 + 0 \times 0 \times 1 +$$
$$1 \times 1 \times 1 + 0 \times 1 \times 1 + 1 \times 1 \times 1 + 0 \times 1 \times 1 + 0 \times 1 \times 1$$
$$= 3$$

5

Document 3:

$$E (3) = 1 \times 1 \times 1 + 0 \times 1 \times 1 + 1 \times 1 \times 1 + 1 \times 1 \times 1 +$$
$$1 \times 1 \times 1 + 0 \times 1 \times 1 + 1 \times 1 \times 1 + 1 \times 1 \times 1 +$$
$$0 \times 1 \times 1 + 0 \times 1 \times 1 + 0 \times 1 \times 1$$
$$= 6$$

As a result, the ranking is: document 3 - document 1 - document 2.

Fig. 7 shows examples of the document ranking table in the above case. In Fig. 7, subtotal of the scores of document feature of each document is given as partial score, and the same applies to all of the subsequent ranking tables. Although it is not given in Fig. 7, frequency of appearance of the document feature in each document or weight given to the document feature classification and the document feature can be given in the ranking table.

Also, ranking can be given by putting weight to the input feature classification and the input feature. For example, to the examples of the data of the input feature classification storage unit 17 shown in Fig. 5, it is supposed that the user specifies, via the user request processing unit 12, thus collating with the documents having the document feature classification and the document feature shown in Fig. 6, presuming that the weight of the input feature classification is 1, the weight of the input feature belonging to the input feature classification C is 1, and the weight of the other input feature classification and the input feature is 0.

Fig. 8 shows examples of the data of the corrected input feature classification storage unit 17 as the result of the weighting to the data of the input feature classification storage unit of Fig. 5 at the request of the user.

In the data of the input feature classification storage unit 17 of Fig. 8, the frequency of the input feature is 1 in all cases, and description is omitted. The same applies to the explanation of the data of the input feature classification storage unit 17 hereinafter. It is supposed that weight is given to the document feature classification and the document feature according to:

Rule 2:

(2a) to the document feature classifications having corresponding input feature classification, give weight of the corresponding input feature classification;

(2b) to the document features having corresponding input feature, give weight of the corresponding input feature; and

(2c) to give weight 0 to all of the document feature classifications and the document features other than the (1a) and (1b) above.

Because the in-document frequency of the document feature of each document is 1 as given in Fig. 6, when the score of each document is calculated by the calculation method shown in the above equation 1, the ranking of the documents 1 to 3 is given as: document 3 - document 2 - document 1.

Fig. 9 summarizes an example of document ranking table obtained as the result. In Fig. 9, the document feature classification and the document feature with weighting are enclosed by the symbol "[ ]".

Also, it is possible to give ranking of documents, supposing that the document feature classification and the document feature as provisional input feature classification and input feature. For example, it is supposed that the user requests to collate, among the document features belonging to the document feature classification C in the documents 2 and 3 in Fig. 6, the document features "boil", "steam" and "dress" as provisional input features with the documents of Fig. 6 and that this has been input via the user request processing unit 12. Fig. 10 shows examples of the provisional input feature classification and the input features to be stored in the input feature classification storage unit 17 in case the weight of the provisional input feature classification C is 1 and weight of each of the provisional input features "boil", "steam" and "dress" is 1 respectively.

The weights of the document feature classification and the document feature are determined according to the above Rule 2. When the ranking of the documents is calculated using the evaluation function of the equation 1, supposing that the indocument frequency of the document feature of each document is 1, the ranking is: document 3 - document 2 - document 1. The ranking table thus obtained is given in Fig. 11.

The user can add the document feature classification having no corresponding input feature classification or the document feature having no corresponding input feature, as an input feature classification or an input feature. For example, it is supposed that the user inputs a request via the user request processing unit 12, that the document features "boil", "steam" and "dress" belonging to the document feature classification C of the documents 2 and 3 in Fig. 6 should be added to the input feature classification C of Fig. 5. Because the above three input features specified by the user are

not included in the input feature classification C, among the document features of the document feature classification C, these are added to the input feature classification storage unit 17 as new input features of the input feature classification C.

Further, in case the user inputs, via the user request processing unit 12, a request that the document feature classification F of the document 2 of Fig. 6 and the document feature "mackerel" and "Spanish mackerel" and input feature classification and its input features, these are added to the input feature classification storage unit 17 as new input feature classification and its input features because there are none to correspond to the document feature classification F and its document features in the input feature classification and the input features of Fig. 5.

Fig. 12 shows an example of data of the input feature classification storage unit 17 after correction, which is obtained by adding the document feature classification F and its document feature and some of the document features among the document feature classification C to the data of the input feature classification storage unit of Fig. 5. In Fig. 12, the newly added input feature classification and the input features newly added are enclosed by the symbols "* *".

Also, ranking is given by newly adding weight to the corrected input feature classification or the input features. For example, it is supposed that the user sees the data of the input feature classification storage unit 17 corrected as shown in Fig. 12 and wants to retrieve by selecting the document relating to "stewing vegetables or fishes". The user sets the weight of the input feature classifications A and F of Fig. 12 as 5, the weight of input feature belonging to the input feature classifications A or F as 1, the weight of the input feature classification C as 5, the weight of the feature "stew" among the input features belonging to the input feature classification C as 10, the weight of the input features other than "stew" belonging to the input feature classification C as 0, and the weight of the other input feature classifications and the input features as 0. Fig. 13 shows the data of the input feature classification storage unit 17 thus corrected.

The weight of the document feature classification and the document feature are determined according to the above Rule 2. If it is supposed that the in-document frequency of the document feature of each document is 1, the ranking of the documents is calculated using the evaluation function of the equation 1, and the ranking is: document 2 - document 3 - document 1. Fig. 14 shows the document ranking table thus obtained.

As described above, according to the present invention, the contents of query and the document are presented to the user as corresponding relationship between the partial concept expressing each of the contents and linguistic features belonging to each concept. If necessary, the user performs collating by adding correction and weighting to each of the concepts and linguistic features, and the collating method and the result of collating are easily confirmed. As a result, it is possible to obtain advantageous effects, i.e. to efficiently select the document closer to the user's intention of retrieval and to perform retrieval with high accuracy.

It should be understood that the foregoing relates to only preferred embodiments of the present invention, and that it is intended to cover all changes and modifications of the embodiments of the invention herein used for the purpose of the disclosure which do not departs from the spirit of the invention.

Linguistic features extracted from the query are classified into concepts expressing content of the query, and linguistic features extracted from the document are classified into concepts expressing content of the document. The user confirms of which concept the input statement and the document are comprised and which kind of linguistic feature corresponds to each concept, and the system assists the user to adequately select a document, which is closer to the intention of retrieval.

Claims

1. A document retrieval system, comprising:

an input/output control unit for receiving input from a user and for presenting result of processing to the user;
a user request processing unit for accepting request other than query among the inputs from the user, and for processing content of the request;
a document storage unit for storing a document to be retrieved;
a feature extracting unit for extracting linguistic feature of the input statement inputted from said input/output control unit and for extracting linguistic feature of the document stored in said document storage unit as document feature;
an input feature storage unit for storing input feature extracted from the query by said feature extracting unit;
a document feature storage unit for storing linguistic feature taken out of the document by said feature extracting unit;
a feature classifying unit for classifying the input features stored in said input feature storage unit so that it corresponds to concept of the content expressed by the query, or for classifying the document feature stored in said document feature storage unit so that it corresponds to concept of the contents expressed by the document;
an input feature classification storage unit for storing correspondence of the input feature classification and the

input feature as a result of classification of the input feature by said feature classification unit; and

a document feature classification storage unit for storing correspondence of the document feature classification with the document feature as a result of classification of the document feature by said feature classifying unit, whereby:

in response to a request sent from the user via the user request processing unit, the contents of the query or the specified document are presented to the user via said input/output control unit as corresponding relationship of group of concepts expressed by the input feature classification and the document feature classification with the linguistic feature belonging to each concept group.

2. A document retrieval system according to Claim 1, wherein, in response to a request of correction from the user via the user request processing unit, correspondence of the query with the input feature classification stored in the input feature classification storage unit, the input feature classification and the input feature, and correspondence of the accumulated document with the document feature classification storage in the document feature classification storage unit and the document feature classification and the document feature as well as the document feature stored in the document feature storage unit are corrected.

3. A document retrieval system according to Claim 1 or 2, wherein there are provided a feature collating unit for collating the input feature classification stored in the input feature classification storage unit and the corresponding input feature with the document feature classification stored in the document feature classification storage unit and the corresponding document feature, and a collating result storage unit for storing method and result of collating by said feature collating unit.

4. A document retrieval system according to Claim 3, wherein, when the input statement is collated with the accumulated document, a specific document feature classification and document feature selected by the user via the user request processing unit are treated as provisional input feature classification and input feature, and collated with the document feature classification and the document feature of each document by the feature collating unit, and the result of the collating is presented to the user.

5. A document retrieval system according to Claim 3 or 4, wherein, when the query is collated with the accumulated document, weight is set to the input feature classification and the input feature depending upon the degree of importance specified by the user via the user request processing unit, and said weighted input feature classification or input feature are collated with the document feature classification or the document feature of each document by said feature collating unit, and the result of the collating is presented to the user.

10/16/2007, EAST Version: 2.0.3.0

FIG. 1
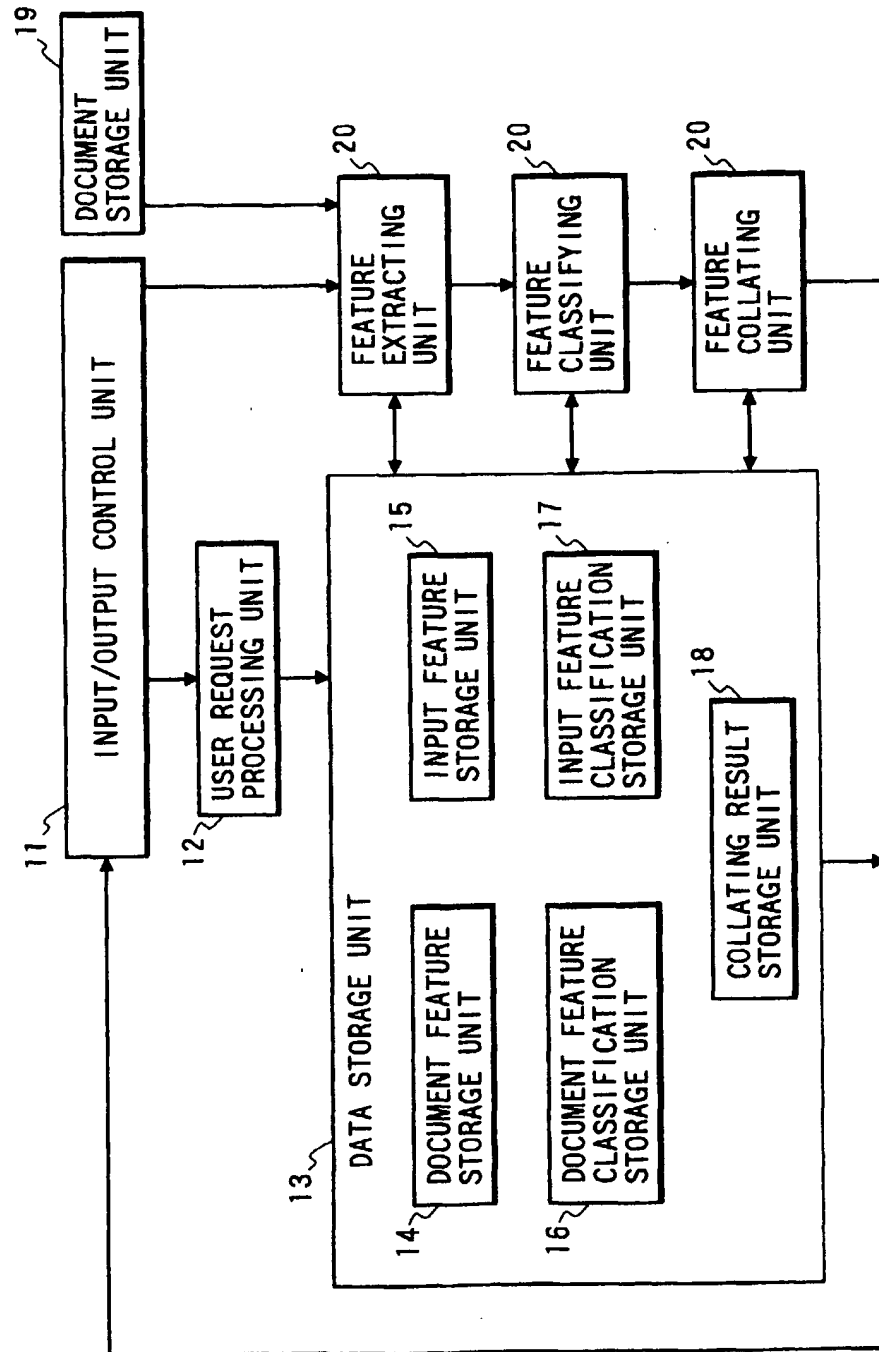
FIG. 2

RADISH, CARROT, POTATO OR KIDNEY BEANS ARE BOILED USING
SINGLE-HANDLE OR TWO-HANDLE POT, OR BAKED OR FRIED USING
FRYING PAN.

FIG. 3

INPUT FEATURE	FREQUENCY
RADISH	1
CARROT	1
POTATO	1
KIDNEY BEANS	1
SINGLE-HANDLE POT	1
TWO-HANDLE POT	1
STEW	1
FRYING PAN	1
BAKE	1
FRY	1

FIG. 4

DEPTH
1
    VEGETABLE

2   GREENISH YELLOW VEGETABLE   LIGHT COLOR VEGETABLE   BEANS

3   CARROT PUMPKIN PIMENTO   RADISH POTATO CABBAGE   SOY BEANS KIDNEY BEANS BROAD BEANS

DEPTH
1   COOKING UTENSILS   COOKING METHOD

2   POT FRYING PAN CHOPPING BOARD   STEW BAKE FRY STEAM

3   TWO-HANDLE POT SINGLE-HANDLE POT
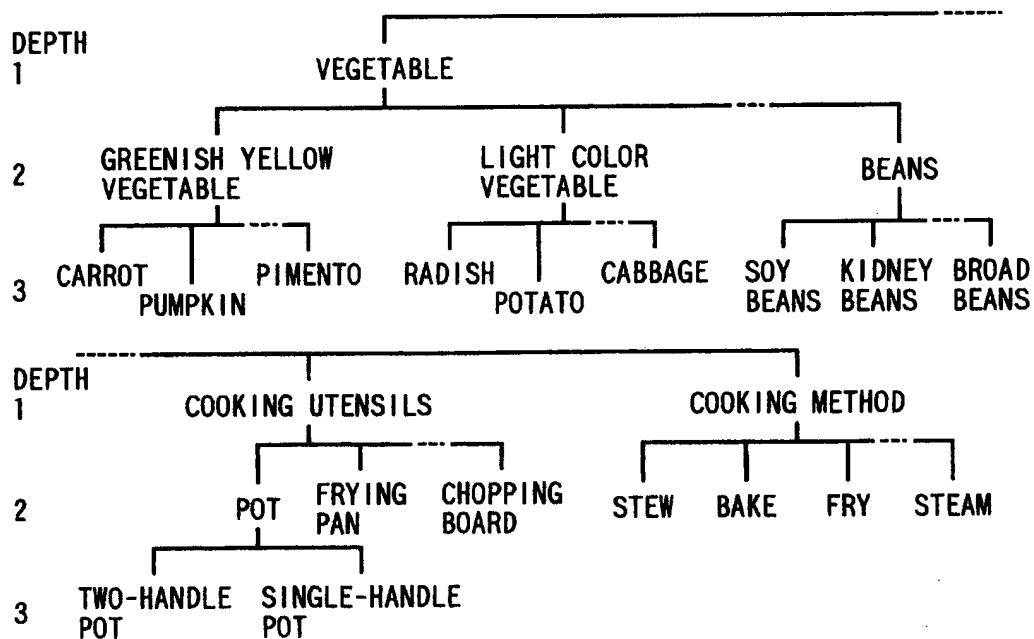
FIG. 5

INPUT FEATURE CLASSIFICATION	INPUT FEATURE	FREQUENCY
A : VEGETABLE	RADISH	1
	CARROT	1
	POTATO	1
	KIDNEY BEANS	1
B : COOKING UTENSILS	SINGLE-HANDLE POT	1
	TWO-HANDLE POT	1
	FRYING PAN	1
C : COOKING METHOD	STEW	1
	BAKE	1
	FRY	1

FIG. 6

DOCUMENT	DOCUMENT FEATURE CLASSIFICATION	DOCUMENT FEATURE	FREQUENCY
1	A : [VEGETABLE]	[RADISH]	1
		[CARROT]	1
		[POTATO]	1
		[KIDNEY BEANS]	1
	D : HORTICULTURE TOOLS	SHOVEL	1
		BUCKET	1
		PLANTER	1
		WATERING POT	1
	E : CULTIVATION	PLANT	1
		SOW	1
		DIG	1
2	A : [VEGETABLE]	[RADISH]	1
		SPINACH	1
	F : FISH	MACKEREL	1
		SPANISH MACKEREL	1
	B : [COOKING UTENSILS]	[SINGLE-HANDLE POT]	1
		BASKET	1
	C : [COOKING METHOD]	[STEW]	1
		BOIL	1
		STEAM	1
3	A : [VEGETABLE]	[POTATO]	1
		SPINACH	1
	B : [COOKING UTENSILS]	[SINGLE-HANDLE POT]	1
		[FRYING PAN]	1
		[TWO-HANDLE POT]	1
		BASKET	1
	C : [COOKING METHOD]	[STEW]	1
		[FRY]	1
		BOIL	1
		STEAM	1
		DRESS	1

FIG. 7

RANKING	DOCUMENT	SCORE	DOCUMENT FEATURE CLASSIFICATION	PARTIAL SCORE	DOCUMENT FEATURE
[1]	3	6	B : [COOKING UTENSIL]	3	[SINGLE-HANDLE POT], [FRYING PAN], [TWO-HANDLE POT], BASKET
			C : [COOKING METHOD]	2	[STEW], [FRY], BOIL, STEAM, DRESS
			A : [VEGETABLE]	1	[POTATO], SPINACH
[2]	1	4	A : [VEGETABLE]	4	[RADISH], [CARROT], [POTATO], [KIDNEY BEANS]
			D : HORTICULTURE TOOLS	0	SHOVEL, BUCKET, PLANTER, WATERING POT
			E : CULTIVATION	0	PLANT, SOW, DIG
[3]	2	3	A : [VEGETABLE]	1	[RADISH], SPINACH
			B : [COOKING UTENSIL]	1	[SINGLE-HANDLE POT], BASKET
			C : [COOKING METHOD]	1	[STEW], BOIL, STEAM
			F : FISH	0	MACKEREL, SPANISH MACKEREL

FIG. 8

INPUT FEATURE CLASSIFICATION	WEIGHT OF INPUT FEATURE CLASSIFICATION	INPUT FEATURE	WEIGHT OF INPUT FEATURE
A : VEGETABLE	0	RADISH	0
		CARROT	0
		POTATO	0
		KIDNEY BEANS	0
B : COOKING UTENSILS	0	SINGLE-HANDLE POT	0
		TWO-HANDLE POT	0
		FRYING PAN	0
C : COOKING METHOD	1	STEW	1
		BAKE	1
		FRY	1

14

FIG. 9

RANKING	DOCUMENT	SCORE	DOCUMENT FEATURE CLASSIFICATION	PARTIAL POINT	DOCUMENT FEATURE
[1]	3	2	C : [COOKING METHOD]	2	[STEW], [FRY], BOIL, STEAM, DRESS
			A : VEGETABLE	0	POTATO, SPINACH
			B : COOKING UTENSIL	0	SINGLE-HANDLE POT, FRYING-PAN, TWO-HANDLE POT, BASKET
[2]	2	1	C : [COOKING METHOD]	1	[STEW], BOIL, STEAM
			A : VEGETABLE	0	RADISH, SPINACH
			B : COOKING UTENSIL	0	SINGLE-HANDLE POT, BASKET
			F : FISH	0	MACKEREL, SPANISH MACKEREL
[3]	1	0	A : VEGETABLE	0	RADISH, CARROT, POTATO, KIDNEY BEANS
			D : HORTICULTURE TOOLS	0	SHOVEL, BUCKET, PLANTER, WATERING POT
			E : CULTIVATION	0	PLANT, SOW, DIG

FIG. 10

PROVISIONAL INPUT FEATURE CLASSIFICATION	WEIGHT OF PROVISIONAL INPUT FEATURE CLASSIFICATION	PROVISIONAL INPUT FEATURE	WEIGHT OF PROVISIONAL INPUT FEATURE
C : COOKING METHOD	1	BOIL	1
		STEAM	1
		DRESS	1

16

FIG. 11

RANKING	DOCUMENT	SCORE	DOCUMENT FEATURE CLASSIFICATION	PARTIAL POINT	DOCUMENT FEATURE
[1]	3	3	C : [COOKING METHOD]	3	STEW, FRY, [BOIL], [STEAM], [DRESS]
			A : VEGETABLE	0	POTATO, SPINACH
			B : COOKING UTENSIL	0	SINGLE-HANDLE POT, FRYING-PAN, TWO-HANDLE POT, BASKET
[2]	2	2	C : [COOKING METHOD]	2	STEW, [BOIL], [STEAM]
			A : VEGETABLE	0	RADISH, SPINACH
			B : COOKING UTENSIL	0	SINGLE-HANDLE POT, BASKET
			F : FISH	0	MACKEREL, SPANISH MACKEREL
[3]	1	0	A : VEGETABLE	0	RADISH, CARROT, POTATO, KIDNEY BEANS
			D : HORTICULTURE TOOLS	0	SHOVEL, BUCKET, PLANTER, WATERING POT
			E : CULTIVATION	0	PLANT, SOW, DIG

FIG. 12

INPUT FEATURE CLASSIFICATION	INPUT FEATURE
A : VEGETABLE	RADISH, CARROT, POTATO, KIDNEY BEANS
B : COOKING UTENSILS	SINGLE-HANDLE POT, TWO-HANDLE POT, FRYING PAN
C : COOKING METHOD	STEW, BAKE, FRY, *BOIL*, *STEAM*, *DRESS*
F : *FISH*	*MACKEREL*, *SPANISH MACKEREL*

FIG. 13

INPUT FEATURE CLASSIFICATION	WEIGHT OF INPUT FEATURE CLASSIFICATION	INPUT FEATURE	WEIGHT OF INPUT FEATURE
A : VEGETABLE	5	RADISH	1
		CARROT	1
		POTATO	1
		KIDNEY BEANS	1
B : COOKING UTENSILS	0	SINGLE-HANDLE POT	0
		TWO-HANDLE POT	0
		FRYING PAN	0
C : COOKING METHOD	5	STEW	10
		BAKE	0
		FRY	0
		BOIL	0
		STEAM	0
		DRESS	0
F : *FISH*	5	*MACKEREL*	1
		SPANISH MACKEREL	1

10/16/2007, EAST Version: 2.0.3.0

FIG. 14

RANKING	DOCUMENT	SCORE	DOCUMENT FEATURE CLASSIFICATION	PARTIAL POINT	DOCUMENT FEATURE
[1]	2	65	C : [COOKING METHOD]	50	[STEW], BOIL, STEAM
			F : [FISH]	10	[MACKEREL], [SPANISH MACKEREL]
			A : [VEGETABLE]	5	[RADISH], SPINACH
			B : COOKING UTENSIL	0	SINGLE-HANDLE POT, BASKET
[2]	3	55	C : [COOKING METHOD]	50	[STEW], FRY, BOIL, STEAM, DRESS
			A : [VEGETABLE]	5	[POTATO], SPINACH
			B : COOKING UTENSIL	0	SINGLE-HANDLE POT, FRYING PAN, TWO-HANDLE POT, BASKET
[3]	1	20	A : [VEGETABLE]	20	[RADISH], [CARROT], [POTATO], [KIDNEY BEANS]
			D : HORTICULTURE TOOLS	0	SHOVEL, BUCKET, PLANTER, WATERING POT
			E : CULTIVATION	0	PLANT, SOW, DIG

20

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 97 11 3355

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	EP 0 638 870 A (HITACHI LTD) * page 4, line 15 - page 7, line 32; figures 10-20 * * column 14, line 54 - column 17, line 26 * ---	1-5	G06F17/30
A	WO 95 30981 A (HUTSON WILLIAM H) * abstract * * page 4, line 24 - page 9, line 19; figures 1-7 * ---	5	
A	GIGER H P: "CONCEPT BASED RETRIEVAL IN CLASSICAL IR SYSTEMS" PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL. (SIGIR), GRENOBLE, JUNE 13 - 15, 1988, no. CONF. 11, 13 June 1988, CHIARAMELLA Y, pages 275-289, XP000295044 * page 282, line 26 - page 284, line 9; figures 4-1 * ---	1,5	
A	EP 0 704 810 A (HITACHI LTD) * page 4, line 56 - page 5, line 26 * * page 3, line 20 - line 50 * * page 10, line 42 - line 50; figure 21 * -----	1	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	3 November 1997	Deane, E

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

FEATURES: Real-Time Adaptive Feature and Document Learning for Web Search

Zhixiang Chen, Xiannong Meng, and Richard H. Fowler
Department of Computer Science, University of Texas–Pan American, 1201 West University Drive, Edinburg, TX 78539-2999. E-mail: chen@cs.panam.edu, meng@cs.panam.edu, fowler@panam.edu

Binhai Zhu
Department of Computer Science, Montana State University, Bozeman, MT 59717. E-mail: bhz@cs.montana.edu

In this article we report our research on building FEATURES—an intelligent web search engine that is able to perform real-time adaptive feature (i.e., keyword) and document learning. Not only does FEATURES learn from the user's document relevance feedback, but it also automatically extracts and suggests indexing keywords relevant to a search query and learns from the user's keyword relevance feedback so that it is able to speed up its search process and to enhance its search performance. We design two efficient and mutual-benefiting learning algorithms that work concurrently, one for feature learning and the other for document learning. FEATURES employs these algorithms together with an internal index database and a real-time meta-searcher to perform adaptive real-time learning to find desired documents with as little relevance feedback from the user as possible. The architecture and performance of FEATURES are also discussed.

1. Introduction

As the World Wide Web rapidly evolves and grows, web search has come to provide an interface between the human users and the vast information on the web in people's daily life. There have been a number of popular and successful general-purpose or meta search engines such as AltaVista [a], Yahoo! [b], Google [c], MetaCrawler [d], Dogpile [e], and Inference Find [f]. Many of the existing engines support personalization (or customization) with the help of pre-defined user profiles or a collection of customizable parameters such as suggestions about keywords to include or exclude, language choices, document locations, etc. These functions can help a search engine find more relevant documents for the user. User profiles are often automatically created by means of *cookies*, client-side *digital traces* or tracking of user's browsing patterns (Bollacker, Lawrence, & Giles, 1999; Widyantoro, Ioerger, & Yu, 1999; Meng & Chen, 1999), or manually created by users themselves. Profiles can be used at either the server side or client side. In most cases, the collection of customizable parameters is fixed. In other cases, some of those parameters can be automatically generated by tracking the recent browsing processes of a user. For example, a search engine can compile a list of suggested keywords based on its internal ranking and the user's most recent browsing contents for use in future search. In essence, the nature of the personalization (or customization) are *static* in the sense that it is defined before a search process and is not able to support real-time adaptive learning from the user's relevance feedback through interactive refinements.

One approach for the next generation of intelligent search engines is that they be built on top of existing search engine design and implementation techniques. They may be built by integrating intelligent components with one general-purpose search engine or with a collection of general-purpose search engines through meta-searching. An intelligent search engine would use the search results of the general-purpose search engines as its starting search space, from which it would adaptively learn from the user's feedback to boost and to enhance the search performance and the relevance accuracy. It may use feature extraction, document clustering and filtering, and other methods to help an adaptive learning process. Recent research on web communities (Kleinberg, 1999; Gibson, Kleinberg, & Raghavan, 1998; Chakrabarti et al., 1998) has used a short list of web pages returned by a search engine as a starting set for further expansion of search. There has been considerable effort applying machine learning to web search-related applications, e.g., scientific article locating and user profiling (Bol-

lacker, Lawrence, & Giles, 1998; 1999; Lawrence, Bollacker, & Giles, 1999), focused crawling (Rennie & McCallum, 1999), collaborative filtering (Nakamura & Abe, 1998; Billsus & Pazzani, 1998), and user preference boosting (Freund, Iyer, Schapire, & Singer, 1998). An adaptive real-time search algorithm without an index, which is basically a focused search starting at some given url and crawling within some neighboring documents, is given in Ikeji and Fotouhi (1999).

FEATURES is part of our research on building an intelligent search engine (Chen, Meng, Zhu, & Fowler, 2000; Chen & Meng, 2000). In this article, document features are limited to keywords that are used to index documents, but our approach may be applied to other cases of document features. Given a search engine S, we use two new concepts, dynamic features and dynamic vector space, to explore the search result $R(q, u)$ returned by S for any query q and any user u. Our strategy is that we use the dynamic features that are relevant to the query q to map the whole document search space to a substantially smaller subspace, the dynamic vector space, that is relevant to the query. We present a feature-learning algorithm that extracts a small set of the dynamic features, i.e., the most relevant indexing keywords to the search query at the moment, and suggests those keywords to the user for him/her to judge whether they are indeed relevant or not. The feature-learning algorithm works concurrently with the document-learning algorithm operating on relevance judgments to retrieve relevant documents for the user. Both learning algorithms help each other to speed up the search process and enhance search performance. An internal index database is used in which each document is indexed using about 300 keywords. We also design and implement a meta-searcher for FEATURES through real-time meta-searching, parsing, and indexing. FEATURES uses an internal index database, a real-time meta-searcher, and learning algorithms to perform real-time adaptive learning for web search to retrieve relevant documents requiring the least possible feedback.

The rest of this article is organized as follows. In section 2, we discuss the necessity of adaptive learning from relevance feedback for web search. In section 3, we introduce two new concepts, the dynamic features and the dynamic vector space, which are relevant to a search query. In section 4, we present the feature-learning and the document-learning algorithms and strategies for feature ranking and document ranking as well as a method for simulating equivalence queries. In section 5, we explain the design and implementation of FEATURES. The performance of FEATURES is also discussed. We conclude the article in section 6.

2. Should We Learn From Relevance Feedback or Should We Not?

The *static* nature of search engines can be understood formally as follows. Let $\mathcal{W}$ denote the collection of web documents, $\mathcal{T}$ denote the set of time values, $\mathcal{Q}$ the set of all

possible queries, and $\mathcal{U}$ the set of all users. Mathematically, a search engine S can be understood as a map f_S as follows:

$$f_S : \mathcal{Q} \times \mathcal{U} \times \mathcal{T} \Rightarrow 2^{\mathcal{W}}.$$

That is, given any query $q \in \mathcal{Q}$, any user $u \in \mathcal{U}$, and any time $t \in \mathcal{T}$, $f_S(q, u, t)$ is a subset of documents in $\mathcal{W}$. We say that a search engine S is static, if

$$f_S(q, u, t) = f_S(q, u', t')$$

for any $q \in \mathcal{Q}$, $u, u' \in \mathcal{U}$, and $t, t' \in \mathcal{T}$ with $|t - t'| \leq B$ for some constant B. As far as we understand, the existing general-purpose search engines (or meta-search engines) are static, because within certain time intervals (say, intervals between updates of the index database or search strategies), the search result of the engine is dependent on the query only.

The static nature of the existing search engines makes it very difficult, if not impossible, to support the *dynamic* changes of the user's search interests. The following scenario is not untypical. Let us suppose that X is a computer scientist and his daily work and web search are all computer science-oriented. However, one day X wanted to search for several good geometry reference books for his child in high school to read. All his personalization (or customization) is heavily computer science-oriented and thus provides no help but misleading search directions. Turning off the augmented features of personalization (or customization) is a simple and nice trick. This time X received *high school*-related pages and *geometry*-related pages. Sadly, the high school-related pages are not related to high school geometry and the geometry related pages are oriented for computer science professionals, because the ranking and search strategy of the search engine is for general purpose.

The augmented features of personalization (or customization) certainly help a search engine to increase its search performance; however, their ability is very limited. An intelligent search engine should be built on top of existing search engine design and implementation techniques. It should use the search result of the general-purpose search engines as its starting search space, from which it would adaptively learn in real-time from the user's relevance feedback to boost and enhance search performance and relevance accuracy. With the ability to perform real-time adaptive learning from relevance feedback, the search engine is able to learn a user's search interest changes or shifts, and thus provides the user with improved search results.

Relevance feedback is the most popular query reformation method in information retrieval (Salton, Wong, & Yang, 1975; Baeza-Yates & Ribeiro-Neto, 1999). It is essentially an adaptive learning process from the document examples judged by the user as relevant or irrelevant. It requires a sequence of iterations of relevance feedback to search for the desired documents. As it is known through the research of Salton, Wong, & Yang (1975) and Salton

(1971), a single iteration of similarity-based relevance feed-back usually produces improvements from 40% to 60% in the search precision, evaluated at certain fixed levels of the recall and averaged over a number of user queries.

Some people have the opinion that web search users are not willing to try iterations of relevance feedback to search for their desired documents. However, the authors think otherwise. It is not a question whether the web search users are not willing to try iterations of relevance feedback to perform their search. It is a question whether we can build an adaptive learning system that supports high search precision increase with several iterations of relevance feedback. The web search users may have no patience to try more than a couple of dozens of iterations of relevance feedback. But, if a system has a 20% or so search precision increase with about five or fewer iterations of relevance feedback, are the users willing to use such a system? The authors believe that the answer will be "yes."

As a matter of fact, the authors conducted a survey about the answers to the above question among a group of 62 students in the summer of 2000, and 96% answered "yes." The survey was conducted as follows. We selected the students in two summer classes as the targeted survey objects. We explained the purpose of the survey. We also explained the concepts of relevance feedback and precision and recall as well. To make sure that the students understood those concepts we demonstrated several examples of the real-time search with relevance feedback using the working prototype of WebSail (Chen et al., 2000) that we have implemented. We then asked the students the following questions: *If a system has a 20% or so search precision increase with one iteration of relevance feedback, do you want to use such a system?* The same question was repeated with the number of iterations being $2, 3, \ldots, 10$, and greater than 10, respectively. We counted how many students answered "yes" and computed the average values. We found that 96% of the students would like to use such a system with five or fewer iterations of relevance feedback if a 20% or so search precision may be achieved. We used the 20% precision increase in our survey because we believe that this increase is reasonable and can be achieved practically.

It is certainly not trivial at all to implement adaptive relevance feedback for web search. Because an adaptive learning process needs to keep the history of the learning, the search engine must use a separate thread for each individual search request and must maintain the thread until the search ends. A search thread usually needs to have its own working space and is very time and space consuming. Imagine that a general-purpose search engine needs to process hundreds or thousands of search requests in every minute. It would be too difficult to maintain so many threads in every single minute and to answer each search request very efficiently. Other difficulties include designing innovative real-time adaptive learning algorithms with as little relevance feedback from the user as possible.

3. Dynamic Features Versus Dynamic Vector Space

In spite of the World Wide Web's size and the high dimensionality of web document indexing features, the traditional vector space model in information retrieval (Salton et al., 1975; Salton, 1989; Baeza-Yates & Ribeiro-Neto, 1999) has been used for web document representation and search. However, to implement real-time adaptive learning with limited computing resource, here, at an Ultra™ One Sun workstation, we cannot apply the traditional vector space model directly. Recall that back in 1998, the Alta-Vista system was running on 20 multiprocessor machines, all of them having more than 130 Giga-Bytes of RAM and over 500 Giga-Bytes of disk space (Baeza-Yates & Ribeiro-Neto, 1999). We need a new model that is efficient enough both in time and space for FEATURES and other web search implementations with limited computing resources. The new model may also be used to enhance the computing performance of a web search system even if enough computing resources are available.

We now examine indexing in web search. Again, in this article, we use keywords as document indexing features. Let X denote the set of all indexing keywords for the whole web (or, practically, a portion of the whole web). Given any web document d, let $I(d)$ denote the set of all indexing keywords in X that are used to index d with nonzero values. Then we have the following two properties:

1. The size of $I(d)$ is substantially smaller than the size of X. Practically, $I(d)$ can be bounded by a constant. The rationale behind this is that in the simplest case we do not need to use all the keywords in d to index it. In our implementation of FEATURES, we use about 300 keywords to index documents in its internal database and at most 64 automatically generated keywords to index a document retrieved through meta-search.
2. For any search process related to the search query q, let $D(q)$ denote the collection of all the documents that match q, then the set of indexing keywords relevant to q, denoted by $F(q)$, is

$$F(q) = \bigcup_{d \in D(q)} I(d).$$

Although the size of $F(q)$ varies from different queries, it is still substantially smaller than the size of X, and might be bounded by a few hundreds or a few thousands in practice.

Definition 3.1.

Given any search query q, we define $F(q)$, which is given in property 2 above, as the set of dynamic features relevant to the search query q.

Definition 3.2.

Given any search query q, the dynamic vector space $V(q)$ relevant to q is defined as the vector space that is

constructed with all the documents in $D(q)$ (as given in property 2 above) such that each of those document is indexed by the dynamic features in $F(q)$.

For any query q, FEATURES first finds the set of documents $D(q)$ that match the query q. It finds $D(q)$ with the help of a general-purpose search strategy through searching its internal database, or through meta-searching AltaVista [a] when no matches are found within its internal database. It then finds the set of dynamic features $F(q)$, and later constructs the dynamic vector space $V(q)$. Once $D(q)$, $F(q)$, and $V(q)$ have been found, FEATURES starts its adaptive learning process from the user's document and feature relevance feedback. More precisely, it uses two learning algorithms in parallel to achieve its goal of learning: One algorithm adaptively learns from the documents judged by the user as relevant or irrelevant examples; the other extracts and suggests a small set of keywords that are most relevant to the search query up to the moment, and learns from the keywords judged by the user as relevant or irrelevant. The feature-learning algorithm helps the document-learning algorithm to increase the ranks of relevant documents, while the document-learning algorithm helps the feature-learning algorithm to increase the ranks of the relevant features.

Example 3.3.

We now give examples to show the significance of our approach of dynamic features and dynamic vector spaces. Here we use an example query "Zhixiang Chen" because it is related to a big collection of web pages due to the popularity of the family name "Chen," and of course, it is related to the web pages of one of authors. On January 12, 2001, we queried the search engine AltaVista with the example query and found out that there are 247,371 related pages. If we follow the traditional vector space approach, then the number of terms (or keywords) used to index those pages would be huge. To get some idea about this number, we simply counted the number of keywords that occurred in all Zhixiang Chen's 267 web pages that reside at the web server of the Computer Science Department, the University of Texas–Pan American. We found 12,347 keywords. When those 247,371 pages are considered, it is obvious that the number of keywords occurred in those pages is much bigger than 12,347. That is, any of those pages would be indexed with many more than 12,347 keywords, or in other words, the dimensionality of the vector space for those pages is much bigger than 12,347.

Now, let us consider dynamic indexing features and dynamic vector space. The top four web pages related to the query "Zhixiang Chen" were

d_1 = www.cs.panam.edu/chen/researchProfile.html
d_2 = cs-people.bu.edu/zchen
d_3 = www.uidaho.edu/micro-biology/faculty/chen.html
d_4 = sunsite.informatik.rwth-aachen.de/dblp/db/indices/
     a-tree/Chen:Zhixiang.html

The dynamic indexing features that were automatically generated by our system for those four pages are

$I(d_1)$ = {algorithm, chen, class, computing, data, edu, field, information, learning, machine, mining, network, neural, new, panam, paper, profile, project, recently, research, retrieval, search, seek, strategies, visualization, web, working, www, zhixiang}

$I(d_2)$ = {american, chen, class, computer, department, edinburg, edu, find, pan, panam, people, profile, research, science, texas, university, usa, zchen, zhixiang}

$I(d_3)$ = {acid, action, biology, chen, class, defense, faculty, home, interest, mechanism, micro, page, plant, regulation, research, response, salicylic, transcriptional, uidaho, www, zhixiang}

$I(d_4)$ = {ameur, bibliography, chen, class, dblp, foued, home, hpsearch, informatik, list, page, publication, rwth, search, server, sunsite, zhixiang}

The set of dynamic features for those four web pages is

F("Zhixiang Chen") = $I(d_1) + I(d_2) + I(d_3) + I(d_4)$ = {acid, action, algorithm, american, ameur, bibliography, biology, chen, class, computer, computing, data, dblp, department, defense, edinburg, edu, faculty, field, find, foued, home, hpsearch, informatik, information, interest, learning, list, machine, mechanism, micro, mining, networks, neural, new, page, pan, panam, paper, people, plant, profile, projects, publication, recently, regulation, research, response, retrieval, rwth, salicylic, science, search, seek, server, strategies, sunsite, texas, transcriptional, university, usa, uidaho, visualization, web, working, www, zchen, zhixiang}

We should point out that the strategy we used to find dynamic features is based on parsing out keywords in the head, title, and the first few sentences of the web pages. A variety of other strategies may be designed, and we plan to study those in the future research. The point is that the number of dynamic features is much smaller than the number of traditional indexing features. Hence, the approach of dynamic features and the vector space is practically feasible for real-time adaptive learning in web search.

4. Feature Learning and Document Learning

As we have investigated (Chen, Meng, & Fowler, 1999; Chen et al., 2000; Chen & Meng, 2000), intelligent web search can be modeled approximately as an adaptive learning process such as online learning (Angluin, 1987; Littlestone, 1988), where the search engine acts as a learner and the user as a teacher. The user sends a query to the engine, the engine uses the query to search the index database, and returns a list of document urls that are ranked according to a ranking function. Then, the user provides relevance feedback, and the engine uses the feedback to improve its next search and returns a refined list of document urls. The learning (or search) process ends when the engine finds the desired documents for the user. Conceptually, a query en-

tered by the user can be understood as the logical expression of the collection of the documents the user wants. A list of document urls returned by the engine can be interpreted as an approximation to the collection of the desired documents.

Rocchio's similarity-based relevance feedback algorithm, one of the most popular query reformation methods of information retrieval (Rocchio, 1971; Ide, 1971; Salton, 1989; Baeza-Yates & Ribeiro-Neto, 1999), is in essence adaptive supervised learning from examples (Salton & Buckley, 1990; Lewis, 1991). We showed (Chen & Zhu, 2000) that for any of the four typical similarity measures (inner product, cosine coefficient, dice coefficient, and Jaccard coefficient) listed in Salton (1989), Rocchio's similarity-based relevance feedback algorithm has a worst-case lower bound that is at least linear in the dimensionality of the Boolean vector space. More precisely, we showed that in the n-dimensional Boolean vector space model, if the initial query vector is 0, then when any of the four typical similarity measures (inner product, dice coefficient, cosine coefficient, and Jaccard coefficient) is used, Rocchio's similarity-based relevance feedback algorithm makes at least n mistakes when used to search for a collection of documents represented by a monotone disjunction of at most k relevant features (or indexing terms). When an arbitrary Boolean initial query vector is used, it makes at least $(n + k - 3)/2$ mistakes to search for the same collection of documents. Our linear lower bound holds for arbitrary classification threshold and updating coefficients used at each step of the algorithm. Because the linear lower bound was proved based on the worst case analysis, they *may not* affect the effective applicability of the similarity-based relevance feedback algorithm. On the other hand, the lower bound helps us understand the algorithm well so that we may find new strategies to improve its performance or design new learning algorithms with better performance.

4.1. The General Setting of Learning

For each particular search query q, with the help of certain general-purpose search strategies, FEATURES first finds the three sets; the general matching document set $D(q)$, the dynamic feature set $F(q)$, and the dynamic vector space $V(q)$. Let $F(q) = \{K_1, \ldots, K_n\}$ such that each K_i denotes a dynamic feature (i.e., an indexing keyword). The two learning algorithms of FEATURES maintain a common weight vector $\mathbf{w} = (w_1, \ldots, w_n)$ for dynamic features in $F(q)$. The components of $\mathbf{w}$ have non-negative real values. The feature-learning algorithm uses $\mathbf{w}$ to extract and learn the most relevant features. The document-learning algorithm also uses $\mathbf{w}$ to classify documents in $D(q)$ as relevant or irrelevant.

One should note that during the learning process both learning algorithms update the common weight vector $\mathbf{w}$ concurrently. Of course, both algorithms need to be equipped with efficient ranking functions. Moreover, we also need to provide a good strategy to simulate the equiv-

alence query for the document-learning algorithm, because the user in reality cannot serve as a *real* teacher as modeled in online learning.

Example 4.1.1. We continue example 3.3 with the example query "Zhixiang Chen." For simplicity let us consider the top four pages returned by AltaVista on January 12, 2001. Please recall that AltaVista found a total of 247,371 relevant pages at the time. According to example 3.3, the set of dynamic features for those four web pages is

$$F(\text{"Zhixiang Chen"}) = I(d_1) + I(d_2) + I(d_3) + I(d_4) =$$
{acid, action, algorithm, american, ameur, bibliography, biology, chen, class, computer, computing, data, dblp, department, defense, edinburg, edu, faculty, field, find, foued, home, hpsearch, informatik, information, interest, learning, list, machine, mechanism, micro, mining, networks, neural, new, page, pan, panam, paper, people, plant, profile, projects, publication, recently, regulation, research, response, retrieval, rwth, salicylic, science, search, seek, server, strategies, sunsite, texas, transcriptional, university, usa, uidaho, visualization, web, working, www, zchen, zhixiang}

This means that we need to use a common weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_{64})$ for computing the weights of all the 68 dynamic features. For example, w_1 represents the weight of "acid," and w_{44} represents the weight of "publication." Remember that the features are sorted in lexicographical order.

4.2. The Feature Learning Algorithm FEX (Feature EXtraction)

For any dynamic feature $K_i \in F(q)$ with $1 \leq i \leq n$, we define the rank of K_i as

$$h(K_i) = h_0(K_i) + w_i.$$

$h_0(K_i)$ is the initial rank for K_i. Recall that K_i is some indexing keyword. With the feature ranking function h and common weight vector $\mathbf{w}$, FEX extracts and learns the most relevant features as follows.

Algorithm FEX. At stage $s \geq 0$, it first sorts all the dynamic features in $F(q)$ with the ranking function h and extracts 10 top-ranked features and suggests them to the user for her to judge their relevance to the query q. When it receives the feature relevance feedback from the user, then for each feature K_i judged by the user as relevant, it promotes K_i by setting $w_i = pw_i$; for each feature judged by the user as irrelevant it demotes it by setting $w_i = w_i/d$. Here, both p and d are, respectively, feature promotion and demotion parameters, and they are tunable.

Example 4.2.1. We continue example 4.1.1. The feature learning algorithm FEX will use features judged by the user

to promote or demote their weights. For example, when the feature"acid" is judged by the user as relevant, then its weight w_1 will be promoted and hence the score of the web page d_3 will be increased. When the feature "publication" is judged by the user as irrelevant, its weight w_{44} will be demoted and hence the score of the web page d_4 will be decreased.

4.3. The Document-Learning Algorithm TW2

The algorithm TW2, a tailored version of Winnow2 (Littlestone, 1988), was designed and successfully implemented in our recent projects WebSail and Yarrow (Chen et al., 2000; Chen & Meng, 2000). Winnow2 sets all initial weights to 1, but TW2 sets all initial weights to 0 and has a different promotion strategy accordingly. Another substantial difference between TW2 and Winnow2 is that TW2 accepts document examples that may not contradict its current classification to promote or demote its weight vector. One must notice that Winnow2 only accepts examples that contradict its current classification to perform promotion or demotion. The rationale behind setting all the initial weights to 0 is not as simple as it looks. The motivation is to focus attention on the propagation of the influence of the relevant documents, and use irrelevant documents to adjust the focused search space. Moreover, this approach is computationally feasible because existing effective document-ranking mechanisms can be coupled with the learning process.

Algorithm TW2 (The tailored Winnow2). TW2 uses the common weight vector w and a real-valued threshold θ to classify documents in $D(q)$. Initially, all weights have value 0. Let $\alpha > 1$ be the promotion and demotion factor. TW2 classifies documents whose vectors $x = (x_1, \ldots, x_n)$ satisfy $\sum_{i=1}^n w_i x_i > \theta$ as relevant, and all others as irrelevant. If the user provides a document that contradicts the classification of TW2, then we say that TW2 makes a mistake. Let $w_{i,b}$ and $w_{i,a}$ denote the weight w_i before the current update and after, respectively. When the user responds with a document which may or may not contradict to the current classification, TW2 updates the weights in the following two ways

- Promotion: For a document judged by the user as relevant with vector $x = (x_1, \ldots, x_n)$, for $i = 1, \ldots, n$, set

$$w_{i,a} = \begin{cases} w_{i,b}, & \text{if } x_i = 0, \\ \alpha, & \text{if } x_i = 1 \text{ and } w_{i,b} = 0, \\ \alpha w_{i,b}, & \text{if } x_i = 1 \text{ and } w_{i,b} \neq 0. \end{cases}$$

- Demotion: For a document judged by the user as irrelevant with vector $x = (x_1, \ldots, x_n)$, for $i = 1, \ldots, n$, set $w_{i,a} = w_{i,b}/\alpha$.

In contrast to the linear lower bounds proved for Rocchio's similarity-based relevance feedback algorithm (Chen & Zhu, 2000), the above learning algorithm has surprisingly small mistake bounds for learning any collection of documents represented by a disjunction of a small number of relevant features. The mistake bounds are independent of the dimensionality of the indexing features. For example

- To learn a collection of documents represented by a disjunction of at most k relevant features (or indexing keywords) over the n-dimensional boolean vector space, TW2 makes at most $[\alpha^2 A/(\alpha - 1)\theta] + (\alpha + 1)k \ln_\alpha \theta - \alpha$ mistakes, where A is the number of dynamic features occurred in the learning process.
- When, on average, l out of k relevant features (or indexing keywords) appear as dynamic features for any relevant document above is judged by the user during the learning process, the bound in Theorem 4.3.1 is improved to $\alpha^2 A/(\alpha - 1)\theta + [(\alpha + 1)^k/l] \ln_\alpha \theta - \alpha$ on average, where A is the number of dynamic features occurred in the learning process.

The actual implementation of the learning algorithm TW2 requires the help of document ranking and equivalence query simulation given in the following two subsections.

Example 4.3.1. We continue example 4.1.1. The document learning algorithm TW2 will use document examples judged by the user to promote or demote the weights of the dynamic features in those examples. For example, when the document d_3 is judged as relevant by the user, then the algorithm TW2 will promote the weights for all dynamic features in $I(d_3) = \{$acid, action, biology, chen, class, defense, faculty, home, interest, mechanism, micro, page, plant, regulation, research, response, salicylic, transcriptional, uidaho, www, zhixiang$\}$. Thus, any document with dynamic features in $I(d_3)$ will also be promoted. When the document d_1 is judged as irrelevant by the user, then the algorithm TW2 will demote the weights for all dynamic features in $I(d_1) = \{$algorithm, chen, class, computing, data, edu, field, information, learning, machine, mining, network, neural, new, panam, paper, profile, project, recently, research, retrieval, search, seek, strategies, visualization, web, working, www, zhixiang$\}$. Consequently, any document with dynamic features in $I(d_1)$ will be demoted.

4.4. Document Ranking

Let g be a ranking function independent of TW2 and FEX. We define the ranking function f for TW2 as follows. For any web document $d \in D(q)$ with vector $x_d = (x_1, \ldots, x_n) \in V(q)$

$$f(d) = \gamma_d[g(d) + \beta_d] + \sum_{i=1}^n w_i x_i,$$

g remains constant for each document d during the learning process of TW2. Various strategies can be used to define g; e.g., PageRank (Brin & Page, 1998), classical tf-idf scheme, vector spread, or cited-based rankings (Yuwono & Lee,
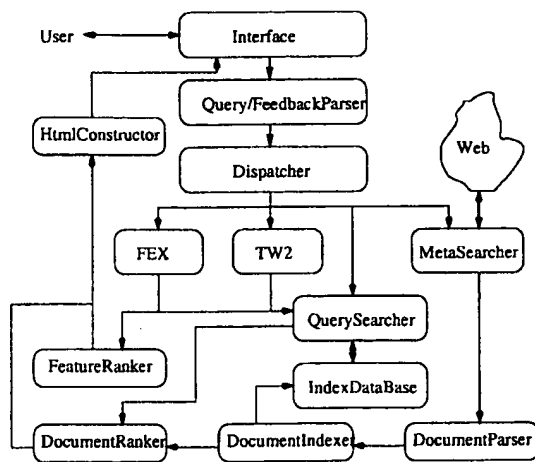
FIG. 1. Architecture of FEATURES.

1996). The two additional tuning parameters are used to do individual document promotions or demotions of the documents that have been judged by the user. Initially, let $\beta_d \geq 0$ and $\gamma_d = 1$. γ_d and β_d can be updated in the similar fashion as w_i is updated by TW2.

4.5. Equivalence Query Simulation

The DocumentRanker of FEATURES uses the ranking function f to rank the documents, and the HtmlConstructor returns the top-10–ranked documents to the user. These top-10–ranked documents represent an approximation to the classification made by TW2. The quantity 10 can be replaced by, say, 25 or 50. But it should not be too large for two reasons: The user may only be interested in a very small number of top-ranked documents, and the display space is limited for visualization. The user can examine the short list of documents and can end the search process, or if some documents are judged misclassified, document relevance feedback can be provided. Sometimes, in addition to the top-10–ranked documents, the system may also provide the user with a short list of other documents below the top 10. Documents in the second short list may be selected randomly. The motivation for the second list is to give the user some better view of the classification made by the learning algorithm.

5. The FEATURES

5.1. The Architecture

FEATURES is implemented on an Ultra One Sun Workstation using 27 Giga-bytes hard disk storage on an IBM R6000 workstation. It is a multithreaded program coded in C++. Its architecture is shown in Figure 1. FEATURES maintains an internal index database with about 834,000 documents, each of which is indexed with about 300 indexing keywords. Besides its internal index database, it has a

MetaSearcher that queries AltaVista when needed. The documents retrieved through meta-search are parsed and indexed by the DocumentParser and the DocumentIndexer that work in real-time. The two learning algorithms FEX and TW2 update the common weight vector w concurrently. The major components and their functions are explained in the next subsection.

5.2. How FEATURES Works

FEATURES has an interface as shown in Figure 2. Using this interface, the user can enter a query and specify the number of document urls to be returned. Having entered query information, she then starts FEATURES. FEATURES invokes its Query/FeedbackParser to parse the query information, document-relevance feedback, or feature-relevance feedback. Then, Dispatcher decides whether the current task is an initial search process or a learning process. If it is an initial search process, Dispatcher first calls QuerySearcher to find the relevant documents within its internal index database. The relevant documents found by QuerySearcher are then passed to DocumentRanker, which ranks the documents and sends them to HtmlConstructor. HtmlConstructor finally generates html content to be shown to the user.

If QuerySearcher fails to find any documents relevant to the query within IndexDataBase, FEATURES calls its MetaSearcher to query AltaVista and retrieve a list of documents. The length of the list is determined by the user. Once the list of the top-matched documents is retrieved, FEATURES calls its DocumentParser and DocumentIndexer to parse retrieved documents, collate them, and index them with at most 64 indexing keywords. The indexing keywords are automatically extracted from the retrieved documents by DocumentParser. The indexed documents will be cached in IndexDataBase and also sent to DocumentRanker and later to HtmlConstructor to be displayed to the user.

Usually, HtmlConstructor shows the top-10–ranked documents, plus the top-10–ranked features, to the user for her to judge document relevance and feature relevance. However, for the initial search, HtmlConstructor shows only the top-ranked documents. The format of presenting the top-10–ranked documents together with the top-10–ranked features is shown in Figure 3. In this format, each document url and each feature are preceded by radio buttons for the user
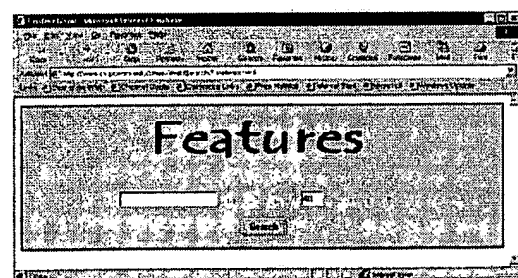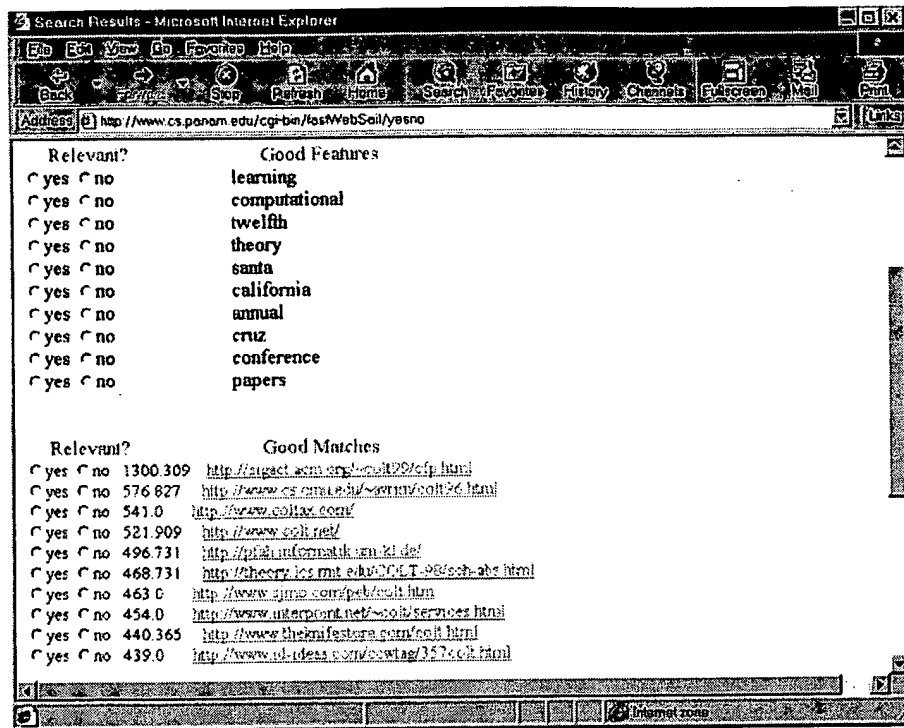

FIG. 2. Interface of FEATURES.

FIG. 3. Initial query result for "colt."

to indicate whether the document or the feature is relevant. The search processes shown in Figures 3 and 4 were performed on March 7, 2000. The query word was "colt" and the desired web documents were those related to "computational learning theory." After two iterations with a total of four relevant and irrelevant documents and five relevant and irrelevant features judged by the user as feedback, all the "colt" related web documents among the initial 100 matched documents were moved to the top-10 positions. The clickable urls could have been selected to view the documents so that the user could make his/her judgment more accurately. After providing relevance feedback, he/she can submit the feedback to FEATURES, view all the document urls, or enter a new query to start a new search process.
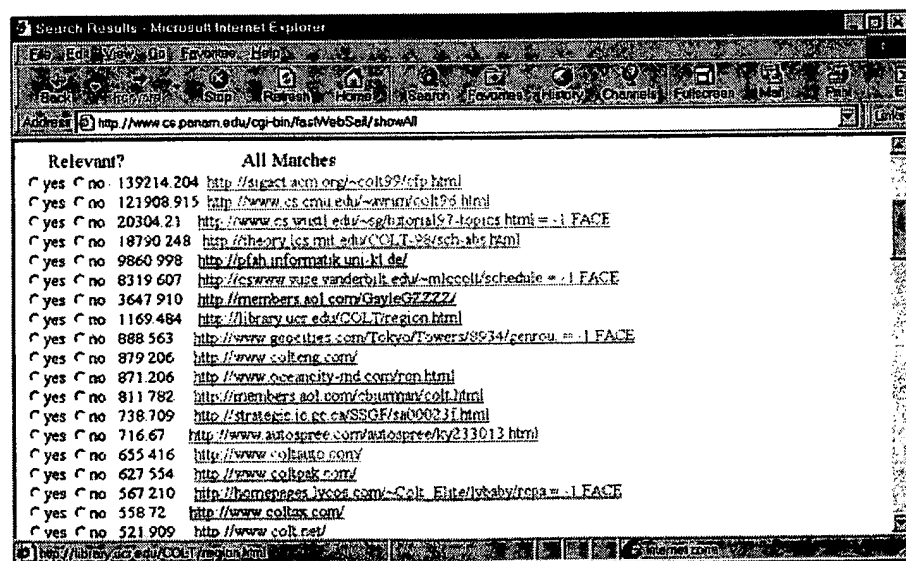
FIG. 4. Result for "colt" after two Iterations, four examples, and five features judged.

If the current task is a learning process from the user's document and feature relevance feedback, Dispatcher sends the feature relevance feedback information to the feature learner FEX and the document relevance feedback information to the document learner TW2. FEX uses the relevant and irrelevant features as judged by the user to promote and demote the related feature weights in the common weight vector w. TW2 uses the relevant and irrelevant documents judged by the user as positive and negative examples to promote and demote the weight vector. TW2 also performs individual document promotion or demotion for those judged documents. Once FEX and TW2 have finished promotions and demotions, the updated weight vector w is sent to QuerySearcher and to FeatureRanker. FeatureRanker re-ranks all the dynamic features that are then sent to HtmlConstructor. QuerySearcher searches IndexDataBase to find the matched documents that are then sent to DocumentRanker. DocumentRanker re-ranks the matched documents and then sends them to HtmlConstructor to select documents and features to be displayed.

Example 5.2.1. When we queried AltaVista with "colt" on March 7, 2000, it found 182,130 relevant pages. For example, the following two pages were among the top-10 pages returned

d_1 = http://sigact.acm.org/cot99/COLT99.html
d_2 = http://www.colteng.com

The dynamic features automatically generated by our system for those two pages are

$I(d_1)$ = {acm, annual, california, class, computational, conference, cruz, july, learning, org, region, registration, santa, sigact, theory, twelfth, university}

$I(d_2)$ = {choice, clients, colteng, com, contractor, corporation, creative, engineering, innovative, people, quote, solutions, striving, welcome}

Obviously, d_1 is relevant to "computational learning theory," while d_2 is not. When d_1 is judged as relevant and d_2 as irrelevant, the document learning algorithm TW2 will promote all the pages with dynamic features in $I(d_1)$ and demote those with dynamic features in $I(d_2)$. After this iteration of the algorithm TW2, dynamic features such as "learning, computational, theory, conference, papers," were among the top-ranked features and then presented to the user to judge for their relevance by the feature learning algorithm FEX. When "learning, computational, theory" were judged by the user as relevant features, all the pages with them as dynamic features would be promoted by the algorithm FEX.

5.3. The Performance: FEATURES Versus AltaVista

We have made FEATURES open for public access. Interested readers can access it via the url given at the end of the

article and check its performance. Although FEATURES is still in its early stages, its actual performance is promising. In order to provide some measure of system performance we have made a comparison between AltaVista [a] and FEATURES. Our evaluation is based on the following approach similar to Recall and Precision.

For a search query q, let A denote the set of documents returned by the search engine (either AltaVista or FEATURES), and let R denote the set of documents in A that are relevant to q. For any integer m with $1 \leq m \leq |A|$, define R_m to be the set of documents in R that are among the top-m–ranked documents according to the search engine. Now, we define the relative Recall R_{recall} and the relative Precision $R_{precision}$ as follows

$$R_{recall} = \frac{|R_m|}{|R|}$$

$$R_{precision} = \frac{|R_m|}{m}$$

We have conducted experiments for 100 search queries, each of which was sent to both AltaVista and FEATURES. The results returned from the two engines were examined manually to calculate R_{recall} and $R_{precision}$. We selected a query based on the following conditions: It should have a long list (say, more than 1,000) of relevant pages. Its relevant pages should have different "clusters" or "groups." Both conditions are easy to verify through querying AltaVista. For example, the query "colt" has 182,130 relevant pages according to the query made on March 7, 2000. Those pages can divided into a group related to "computational learning theory," a group related to "horses," a group related to "guns," a group related to "corporation and engineering," and so on. It is rather easy to find 100 words satisfying the conditions. Among those we selected are "colt, memory, chip, language, high school geometry, michael jordan, harvard, utpa, architecture." The number of keywords in our queries varies from one to four, and the average number is 1.75. During those experiments, the authors served as users and evaluated the documents returned by the search engines. For each query, the set R of relevant pages is defined and understood by the authors. For example, for the query "colt," the R is defined as the set of documents related to "computational learning theory." For the query "chip," the R is defined as the set of documents related to "computer chips." For the query "harvard," the R is defined as the set of documents related to "Harvard University." Since each query we selected has a huge number of relevant pages, in order to perform the experiments we limit R within a short list of top (say, 100)-ranked documents returned by the search engine. We actually read and examined the contents of all the documents within this short list to find R. We followed the same approach to find the set R_m for each query, i.e., we read and examined the contents of the top-

Table 1. Relative precision.

$R_{precision}$	(50,10)	(50,20)	(100,10)	(100,20)	(150,10)	(150,20)	(200,10)	(200,20)
AltaVista	0.36	0.30	0.36	0.30	0.36	0.30	0.36	0.30
FEATURES	0.48	0.40	0.51	0.44	0.52	0.46	0.52	0.46

m–ranked documents to find those that are indeed relevant to the query.

We summarize the average relative Recall R_{recall} and the average relative Precision $R_{precision}$ in Tables 1 and 2. One may notice that FEATURES has better performance by these measures.

In the tables, each column is labeled by a pair of integers $(|A|, m)$, where $|A|$ is the total number of documents retrieved by the engine for the given search query and m is used to define the relative Recall and Precision. Because of the nonadaptive nature of AltaVista, it is obvious that AltaVista has the same relative Precision for each query when the value of m is the same. The average relative Recall and Precision values are calculated for FEATURES after an average of five interactive refinements.

6. Concluding Remarks and Future Work

Web search has come to people's daily life as the web evolves. Designing practically effective web search algorithms is a challenging task. It calls for innovative methods and strategies from many fields including machine learning. As we pointed out, web search can be understood in some sense as adaptive learning from queries. However, few learning algorithms are ready to be used in web search because of a number of realistic requirements. In general, the fundamental question about any learning algorithm is certainly its applicability to real-world problems. In the particular case of web search, we are interested in learning algorithms that can effectively increase the search performance with a very small number (say, five) of relevance feedback iterations, because users do not have their patience to try too many iterations of learning. We think that the most challenging problem regarding intelligent web search is whether we can build an adaptive learning system that supports high search precision increase with several iterations of relevance feedback from the user.

As part of our efforts toward building an intelligent system for web search, we have designed and implemented FEATURES. It utilizes two adaptive learning algorithms that work concurrently in real-time, one of which extracts and learns the most relevant dynamic features from the user's

feature relevance judgments, and the other which learns the most relevant documents from the user's document relevance judgments. FEATURES is still in its early stages and needs to be improved and enhanced in many aspects.

In the future, we plan to improve the interface of FEATURES. Right now, the system displays the urls of the documents. If the user wants to know the contents of the document, he/she needs to click the url to download the content. We plan to display the url of a document together with a good preview of its content. We also want to highlight those indexing keywords in the preview and allow them to be clickable for feature learning.

We also plan to apply clustering techniques to increase the performance of our system. It is easy to observe that in most cases documents that are relevant to a search query can be divided into a few different clusters or groups. Recall that documents relevant to "colt" can be divided into clusters related to "computational learning theory," "guns," "horse," "corporation engineering," etc. We believe that document clustering techniques such as graph spectral partitioning can be used to reduce the number of the iterations of the learning process and to increase the performance of the system.

At its current stage, FEATURES can only query AltaVista for its meta-search purpose. We plan to expand its meta-search capability to allow parallel queries to several existing search engines. We want to use collaborative recommendation methods to collate the search results of the parallel queries.

Because adaptive learning is incremental and depends on the history of a learning process to improve learning performance, FEATURES creates and maintains a specific thread for each web-search process. When a search process is finished, its related thread will be terminated. It may not be easy to employ an adaptive learning algorithm at a popular web-server side. Because a popular web server may have thousands of user accesses in every minute, innovative thread management methods are needed in order to maintain many threads for individual search processes.

Acknowledgments

Table 2. Relative recall.

R_{recall}	(50,10)	(50,20)	(100,10)	(100,20)	(150,10)	(150,20)	(200,10)	(200,20)
AltaVista	0.37	0.51	0.30	0.42	0.29	0.39	0.29	0.39
FEATURES	0.67	0.96	0.42	0.80	0.37	0.71	0.37	0.67

URL References

[a] AltaVista: www.altavista.com
[b] Yahoo!: www.yahoo.com
[c] Google: www.google.com
[d] MetaCrawler: www.metacrawler.com
[e] Inference Find: www.infind.com
[f] Dogpile: www.dogpile.com
[g] WebSail: www.cs.panam.edu/chen/WebSearch/WebSail.html
[h] Yarrow: www.cs.panam.edu/chen/WebSearch/Yarrow.html
[i] Features: www.cs.panam.edu/chen/WebSearch/Features.html

References

Angluin, D. (1987). Queries and concept learning. Machine Learning, 2, 319–432.

Baeza-Yates, R., Ribeiro-Neto, B. (1999). Modern Information Retrieval. Essex, England: Addison-Wesley.

Billsus, D., & Pazzani, M.J. (1998). Learning collaborative information filters. In J. Shavlik (Ed.), Proceedings of the Fifteenth International Conference on Machine Learning (pp. 46–54). San Francisco, CA: Morgan Kaufman Publishers.

Bollacker, K., Lawrence, S., & Giles, C.L. (1998). Citeseer: An autonomous web agent for automatic retrieval and identification of interesting publications. In Proceedings of the Second International Conference on Autonomous Agents (pp. 116–113). New York: ACM Press.

Bollacker, K., Lawrence, S., & Giles, C.L. (1999). A system for automatic personalized tracking of scientific literature on the web. In Proceedings of the Fourth ACM Conference on Digital Libraries (pp. 105–113). New York: ACM Press.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In Proceedings of the Seventh World Wide Web Conference. Geneva, Switzerland: International World Wide Web Conference Commitee.

Chakrabarti, S., Dom, B., Raghavan, P., Rajagopalan, S., Gibson, D., & Kleinberg, J. (1998). Automatic resource compilation by analyzing hyperlink structure and associated text. In Proceedings of the Seventh World Wide Web Conference (pp. 65–74). The Netherlands: Elsevier Science.

Chen, Z., & Meng, X. (2000). Yarrow: A real-time client site meta search learner. In K. Bollacker (Ed.), Proceedings of the AAAI 2000 Workshop on Artificial Intelligence for Web Search (pp. 12–17). Menlo Park, CA: AAAI Press.

Chen, Z., & Zhu, B. (2000). Some formal analysis of the Rocchio's similarity-based relevance feedback algorithm. In D. Lee & S.-H. Teng (Eds.), Proceedings of the Eleventh International Symposium on Algorithms and Computation, Lecture Notes in Computer Science 1969 (pp. 108–119). New York: Springer-Verlag.

Chen, Z., Meng, X., & Fowler, R.H. (1999). Searching the web with queries. Knowledge and Information Systems, 1, 369–375.

Chen, Z., Meng, X., Zhu, B., & Fowler, R. (2000). Websail: From on-line learning to web search. In Q. Li et al. (Eds.), Proceedings of the 2000 International Conference on Web Information Systems Engineering [the full version will appear in Journal of Knowledge and Information Science, the special issue of WISE'00] (pp. 192–199). Piscataway, NJ: IEEE Press.

Freund, Y., Iyer, R., Schapire, R., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. In Machine Learning: Proceedings of the Fifteenth International Conference.

Gibson, D., Kleinberg, J., & Raghavan, P. (1998). Inferring web communities from link topology. In Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia (pp. 225–234). New York: ACM Press.

Ide, E. (1971). New experiments in relevance feedback. In G. Salton (Ed.), The Smart System—Experiments in Automatic Document Processing (pp. 337–354). Englewood Cliffs, NJ: Prentice-Hall.

Ikeji, A., & Fotouhi, F. (1999). An adaptive real-time web search engine. In Proceedings of the ACM CIKM'99 Workshop on Web Information and Data Management (pp. 12–16). New York: ACM Press.

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. Journal of ACM 46, 604–632.

Lawrence, S., Bollacker, K., & Giles, C.L. (1999). Indexing and retrieval of scientific literature. In Proceedings of the Eighth ACM International Conference on Information and Knowledge Management (pp. 139–146). New York: ACM Press.

Lewis, D. (1991). Learning in intelligent information retrieval. In L. Birnbaum and G. Collins (Eds.), Proceedings of the Eighth International Workshop on Machine Learning (pp. 235–239). San Francisco, CA: Morgan Kaufman Publishers.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning, 2, 285–318.

Meng, X., & Chen, Z. (1999). Personalize web search using information on client's side. In J. Luo, B. Xu, Y. Wang, X. Li, & J. Lu (Eds.), Advances in Computer Science and Technologies (pp. 985–992). Beijing, China: International Academic Publishers.

Nakamura, A., & Abe, N. (1998). Collaborative filtering using weighted majority prediction algorithms. In J. Shavlik (Ed.), Machine Learning: Proceedings of the Fifteenth International Conference.

Rennie, J., & McCallum, A. (1999). Using reinforcement learning to spider the web efficiently. In Proceedings of the Sixteenth International Conference on Machine Learning (pp. 335–343). San Francisco, CA: Morgan Kaufman Publishers.

Rocchio, J. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), The Smart Retrieval System—Experiments in Automatic Document Processing (pp. 313–323). Englewood Cliffs, NJ: Prentice-Hall.

Salton, G. (1971). The performance of interactive information retrieval. Information Processing Letters, 1, 35–41.

Salton, G. (1989). Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Reading, MA: Addison-Wesley.

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science, 41, 288–297.

Salton, G., Wong, A., & Yang, C. (1975). A vector space model for automatic indexing. Communications of ACM, 18, 613–620.

Widyantoro, D., Ioerger, T., & Yu, J. (1999). An adaptive algorithm for learning changes in user interests. In Proceedings of the Eight ACM International Conference on Information and Knowledge Management (pp. 405–412).

Yuwono, B., & Lee, D. (1996). Search and ranking algorithms for locating resources on the world wide web. In S.Y.W. Su (Ed.), Proceedings of the International Conference on Data Engineering (pp. 164–171). Piscataway, NJ: IEEE Press.

WILEY InterScience

> HOME
> ABOUT US
> CONTACT US
> HELP
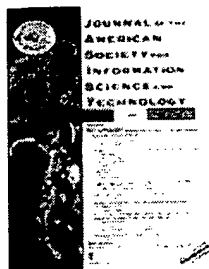
Home / Computer Science / Computer Science (general)

✉ e-mail 🖨 print

Published on behalf of

asis&t

American Society for Information Science and Technology

Go to Society Site

Save Title to My Profile   Set E-Mail Alert

Go to the homepage for this journal to access trials, sample copies, editorial and author information, news, and more. ▸

SEARCH ⦿ All Content
        ○ Publication

Advanced Search
CrossRef / Google Sear
Acronym Finder

SEARCH IN THIS TITL
Journal of the American for Information Science Technology

All Fields

Save Article to My Profile   Download Citation   < Previous Abstract | Next Abstract >

Abstract | References | Full Text: HTML, PDF (160k) | Related Articles | Citation Tracking

Research

FEATURES: **Real-time adaptive feature and document learning for web search**

Zhixiang Chen [1], Xiannong Meng [1], Richard H. Fowler [1], Binhai Zhu [2]

[1]Department of Computer Science, University of Texas-Pan American, 1201 West University Drive, Edinburg, TX 78539-2999
[2]Department of Computer Science, Montana State University, Bozeman, MT 59717

email: Zhixiang Chen (chen@cs.panam.edu) Xiannong Meng (meng@cs.panam.edu) Richard H. Fowler (bhz@cs.montana.edu) Binhai Zhu (bhz@cs.montana.edu)

SEARCH BY CITATION

Vol: Issue: Page:

REPRINT INQUIRIES

Need a reprint?

Paper or electronic repi available for all content published on Wiley Inte Inquiries can be submit online.

Find out more about rej

INDEX TERMS

end user searching • query by example • keyword searching • search engines • machine learning • genetic algorithms

ABSTRACT

In this article we report our research on building **FEATURES** - an intelligent web search engine that is able to perform real-time adaptive feature (i.e., keyword) and document learning. Not only does **FEATURES** learn from the user's document relevance feedback, but it also automatically extracts and suggests indexing keywords relevant to a search query and learns from the user's keyword relevance feedback so that it is able to speed up its search process and to enhance its search performance. We design two efficient and mutual-benefiting learning algorithms that work concurrently, one for feature learning and the other for document learning. **FEATURES** employs these algorithms together with an internal index database and a real-time meta-searcher to perform adaptive real-time learning to find desired documents with as little relevance feedback from the user as possible. The architecture and performance of **FEATURES** are also discussed.

Related Articles

- Find other articles like this in Wiley InterScience
- Find articles in Wiley InterScience written by any of the authors

Wiley InterScience is a member of CrossRef.

crossref Member

Improving browsing in digital libraries with keyphrase indexes

Carl Gutwin [a,*], Gordon Paynter [b,1], Ian Witten [b,2], Craig Nevill-Manning [c,3], Eibe Frank [b,4]

[a] *Department of Computer Science, University of Saskatchewan, 57 Campus Drive, Saskatoon, Saskatchewan, Canada S7N 5A9*
[b] *Department of Computer Science, University of Waikato, Private Bag 3105, Hamilton, New Zealand*
[c] *Department of Computer Science, Rutgers University, Piscataway, NJ 08855, USA*

Abstract

Browsing accounts for much of people's interaction with digital libraries, but it is poorly supported by standard search engines. Conventional systems often operate at the wrong level, indexing words when people think in terms of topics, and returning documents when people want a broader view. As a result, users cannot easily determine what is in a collection, how well a particular topic is covered, or what kinds of queries will provide useful results. We have built a new kind of search engine, Keyphind, that is explicitly designed to support browsing. Automatically extracted keyphrases form the basic unit of both indexing and presentation, allowing users to interact with the collection at the level of topics and subjects rather than words and documents. The keyphrase index also provides a simple mechanism for clustering documents, refining queries, and previewing results. We compared Keyphind to a traditional query engine in a small usability study. Users reported that certain kinds of browsing tasks were much easier with the new interface, indicating that a keyphrase index would be a useful supplement to existing search tools. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Digital libraries; Browsing interfaces; Text mining; Keyphrase extraction; Machine learning

1. Introduction

Digital libraries and searchable document collections are widely available on the Internet. Unfortunately, people still have difficulty finding what they want in these collections (e.g., Refs. [3,13]). In particular, users often find *browsing* to be frustrating. In browsing, the user interacts with a collection and carries out searches, without having in mind a specific document or document set [5]. For example,

one may wish to ascertain what a collection contains, whether there is any material on a certain topic, or what kinds of queries are likely to produce good results (see Table 1). Such tasks are difficult to carry out in digital libraries — not because the problems are inherently hard, but because the search systems that are provided do not support browsing very well.

Standard search engines use a full-text inverted index to find documents that match the user's query, and use a ranking algorithm to order the documents for presentation (e.g., Refs. [35,36]). Although this approach is powerful — users can retrieve a document by entering any word that appears in it — full-text indexing causes several problems when browsing. First, most standard engines index individual words only, whereas people often use short topic

---

* Corresponding author. E-mail: gutwin@cs.usask.ca
[1] E-mail: gwp@cs.waikato.ac.nz.
[2] E-mail: ihw@cs.waikato.ac.nz.
[3] E-mail: nevill@cs.rutgers.edu.
[4] E-mail: eibe@cs.waikato.ac.nz.

Table 1
Three types of browsing and related task questions

Task type	Questions
Collection evaluation	What's in this collection?
	What topics does this collection cover?
Subject exploration	How well does this collection cover area X?
	What topics are available in area X?
Query exploration	What kind of queries will succeed in area X?
	How can I specialize or generalize my query?

phrases when exploring a collection (e.g., *solar energy, programming by demonstration, artificial intelligence*) [5] [38,42]. Second, standard engines return lists of individual documents, but this level of presentation is usually too specific for users who are browsing [34,45]. Third, standard engines do not provide support for forming new queries, even though browsing involves an iterative process of exploration and query refinement [39].

We have built a new search engine — called Keyphind — that uses a single simple approach to address all of these problems. Keyphind is based on keyphrases (similar to those provided by authors) that have been automatically extracted from the documents in a collection. While the use of phrases in information retrieval is not new, Keyphind goes beyond previous systems by making the phrase the fundamental building block for the entire system — including the index, the presentation and clustering of results, and the query refinement process. The keyphrase approach allows people to interact with a document collection at a relatively high level. Topics and subjects are the basic level of interaction, both when forming queries and when viewing and manipulating query results. Browsing is supported by showing the user what topics are available for a particular query, how many documents there are in those areas, and all possible ways that a query can be legitimately extended. In our initial usability studies, the keyphrase approach was seen to be more useful for browsing tasks than a standard query engine.

In the first part of this article, we discuss browsing in more detail, and review previous efforts to address the drawbacks of standard search engines. Then we introduce the Keyphind system, outline the text-mining technique used to extract keyphrases from documents, and describe the ways that users can interact with the system. Finally, we evaluate the keyphrase approach, and report on a usability study that compared browsing in Keyphind with browsing in a traditional full-text query system.

2. Browsing document collections

When people interact with large document collections, only part of their activity involves specific searches for particular documents. Many tasks are better characterized as "browsing" — information-seeking activity where the goal is not a particular set of documents (e.g., Refs. [5,29]). For example, people may want to learn about a collection, what it contains, and how well it covers a particular topic; they may want to find documents in a general area, but without knowing exactly what they are looking for; or they may simply be looking for something "interesting." Browsing is an inherently interactive activity: when people browse, their direction and activities evolve as they work, and they may take several different routes to their final destination.

In bricks-and-mortar libraries, people browse by walking among the stacks, looking at displays set up by library staff, even perusing the reshelving carts to see what other people have taken out. Although digital libraries preclude these physical activities, people still browse; they just do it through whatever means they can — namely the collection's query interface. Browsing in digital libraries is characterized by a succession of queries rather than a single search — that is, people determine their next query partly based on the results of the previous one [29].

This paper focuses on three kinds of browsing tasks that we have informally observed in our experience with the New Zealand Digital Library (NZDL; www.nzdl.org): evaluating collections, exploring topic areas, and exploring possible queries. Table 1 shows some of the questions that people have in mind when they undertake these tasks. The following

---

[5] For example, the typical query to the Infoseek search engine (www.infoseek.com) is a noun phrase with an average length of 2.2 words [23].

sections describe the tasks, outline why traditional search engines make the tasks difficult, and review prior research in each area.

2.1. Collection evaluation

When a searcher has several different document collections to choose from, but not enough time to search all of them in detail, they must determine which collections are most likely to provide them with the information they need. This process of assessment often involves browsing — both to gain an overview of the collection and to determine if it contains material in the area of interest [29]. As an example of the number of collections available for a particular search, the following are a few of the possibilities for finding literature on *browsing interfaces*:

- The ACM Digital Library (www.acm.org/dl)
- Internet search engines (e.g., AltaVista, Infoseek, HotBot)
- The HCI Bibliography (www.hcibib.org)
- The NZDL Computer Science Technical Reports (CSTR) Collection (www.nzdl.org)
- The Cornell Computer Science Technical Reference Library (cs-tr.cs.cornell.edu)
- The INSPEC citation database (local access)
- The NIST TREC document collection (nvl.nist.gov)
- The UnCover citation database (uncweb.carl.org/)
- Local online library catalogues (e.g., library.us-ask.ca).

There are many ways that these and other collections can be compared and evaluated: for example, size, currency, and quality of information are a few possibilities. Perhaps the most important criterion in choosing between collections, however, is coverage — what is in the collection and whether or not it is likely to contain the user's desired information. With physical collections (that is, books), people often evaluate coverage by scanning tables of contents or by looking up their topics in the index (e.g., Ref. [28]).

In digital collections that use standard search engines, however, ascertaining what the collection contains is difficult or impossible. Although most systems provide a brief description of the collection's

contents (e.g., CSTR), they rarely display the range of topics covered. The only way to find this information is by issuing general queries and scanning the lists of returned documents — a slow and arduous process.

Several systems and research projects have sought to address this problem by providing information about the top-level contents of a document collection. These approaches can be split into two groups: those that provide topic information manually, and those that determine it automatically. Manually determined categories involve the classification of each document by a human indexer, and so are laborious to create and maintain. Examples of manual categorization can be seen in commercial systems such as the Yahoo! collection (www.yahoo.com), and in some research collections (e.g., Ref. [14]). In contrast, automatic techniques create high-level overviews by algorithmic examination of the documents in the collection. A variety of techniques for gathering and displaying this information have been investigated. The most well known of these is vector-space document clustering, where document similarities are calculated and used to group similar documents together (e.g., Refs. [26,36]). The process can be repeated until the desired level of generality is reached (although it is not immediately apparent how to choose meaningful names or labels for the clusters [26]). Other researchers have used self-organizing maps to show both the main topics in a collection and their relative frequencies. For example, Lin [28] creates graphical tables of contents from document titles, and Chen and Houston [6] create map representations of a collection of web pages. Finally, when a collection has an internal structure (such as a hierarchical web site), this structure can be used to generate an overview (e.g., Ref. [22]).

2.2. Exploring topic areas

A second common browsing activity is the exploration of a subject area (e.g., Refs. [5,34]). When exploring, users may try to gain an understanding of the topics that are part of the area, may wish to gather contextual information for directing and focusing their search, or may simply hope to come across useful material. Exploration is often undertaken by novices and people who need to learn about

a new area; but it may also be undertaken by people who have had a search failure and are interested in the kinds of queries that will definitely succeed.

This kind of browsing is greatly aided by being able to see the range of materials available about the subject, since people can then both learn about the domain and use visual recognition rather than linguistic recall to direct their activity. As Chang and Rice [5] state, "browsing in computer systems is characterized by searching without specifying; it is a recognition-based search strategy" (p. 243).

Traditional search systems make exploration difficult for two reasons. First, subject areas are often described with multi-word phrases, but the word-level indexes used in many systems deal poorly with queries where query terms are meaningful in relation to one another [43]. For example, a standard system will have poor precision when asked about a subject area like "computer architecture," where the meaning of the phrase is different from either component. Second, ranking algorithms are designed to return documents relevant to the query terms, not to show the range of documents that exist. A list of individual documents is usually far too specific for users who are trying to explore an area (e.g., Refs. [34,39]).

The first of these problems has previously been addressed by including phrases as well as individual words in the inverted index (e.g., Refs. [35,43]). However, indexing all phrases consumes considerable space, so researchers have also considered various means of selecting only "important" or content-bearing phrases for indexing (e.g., Refs. [12,18]). We discuss the issues involved in phrase-based indexing in more detail later in the article.

The second problem — that systems return a list of documents rather than a range of topics — has been addressed primarily through the use of document clustering. In this approach, the search engine returns a set of clusters that match a user's query rather than a set of documents. Again, clustering may be based on existing manual document classifications (e.g., Yahoo) or on automatically extracted information. Automatic approaches are primarily based on vector-space based similarity [26]. Clusters are then presented to the user, either as a list (e.g., Refs. [1,11]) or in some graphical format (e.g., Refs. [19,44]). One of the better-known clustering applications that seeks to support browsing is the Scatter/Gather system [11,33]. This system uses a linear-time vector-space algorithm to dynamically cluster the documents resulting from a user's query. The user can then select one or more clusters for reclustering and further analysis.

2.3. Query exploration

Browsing is an activity where the user's queries are not determined beforehand, but evolve as the user interacts with the collection [5]. Therefore, part of the browsing process involves gathering information about how different kinds of queries will work in the search system, and "translating an information need into a searchable query" (Ref. [3], p. 493). People may wish to find out what kinds of queries will succeed, how a query can be specialized or generalized, or what will happen if an existing query is changed.

In a traditional search system, however, there is little support for query exploration. Users have to compose their queries without any assistance from the system, and there is no indication of how likely a query is to be successful or of what kind of results it will produce. It is also difficult to use a previous query as the basis for a new one: since the inner workings of a ranking algorithm are rarely made explicit, it is not easy to predict how changes to a query will affect the size or relevance of the result set. Compounding the difficulty, a concept may be referred to in the collection by any of several related terms (the vocabulary problem [7,17]), and users must somehow determine the correct one. Without guidance in these areas, users will often issue queries that return either nothing or a multitude of documents — the "feast or famine" problem (e.g., Ref. [14]).

Researchers have attempted to provide guidance for query exploration in several ways. Support for query refinement often presents the user with a set of terms related to their query terms (e.g., Refs. [9,47]); they can then add the suggestions to their query, either disjunctively (to expand the results) or conjunctively (to specialize the results). In some systems, terms are automatically added to the user's query before it is returned, depending on the size of the result set (e.g., Ref. [48]). The related terms may come from a thesaurus, which gathers and records

related terms in the collection (e.g., Ref. [10]), or from other lexical means (e.g., Ref. [1]). Hierarchical and dynamic clustering can also be used to help the user form their next query. For example, the Scatter/Gather system regroups the results every time the user selects a set of clusters to examine further; so at any stage of the search, this always provides the user with a set of possibilities that they know will produce results.

A few systems also provide results previews — feedback about the results of a query before the query is issued (e.g., Refs. [14,20]). These techniques allow users to manipulate query parameters and see qualities of the results set before the query is issued (for example, the number of documents in the set or their distribution across one or more kinds of meta-data). Results previews help users understand which queries are likely to succeed, and enable powerful interaction techniques like dynamic querying [39], but require that a small and fast version of the entire index be available or downloadable to the local query client [20].

2.4. Supporting browsing tasks using phrases

Our approach to supporting these browsing tasks is to use keyphrases as the basic unit for indexing, retrieval, and presentation. Keyphrases provide us with a simple mechanism for supporting collection evaluation, subject-area exploration, and query exploration. The advantages of this approach are that keyphrases can be automatically extracted from documents; that the resulting indexes are simple, small, and quickly built; and that the workings of the system are easy for users to understand. In Section 3, we consider these issues in more detail. We describe the Keyphind system, the technique used to extract keyphrases from documents, and how the keyphrase indexes are built.

3. Keyphind

Keyphind (from "keyphrase find") is a system that permits browsing, exploring, and searching large collections of text documents (see Fig. 1). Like other search engines, it consists of an index and a graphical query interface; however, both index and pre-

sentation schemes differ substantially from conventional systems. First, the index is built from keyphrases that have been automatically extracted from the documents, rather than from the full text of the collection. Second, documents that match a query are grouped by keyphrase in the interface, and users can work with these clusters before looking at individual documents.

More detail on these techniques is given below. First, we take a brief look at what it is like to use the system. Then we describe how it obtains phrases, builds its indexes, and presents documents to the user. The examples and screen snapshots that we discuss use a database built from approximately 26 000 documents, whose source text totals 1 Gb. The reports are part of the CSTR collection of the NZDL [49] (www.nzdl.org). Although the example collection is not enormous by today's standards, the algorithms and techniques used in Keyphind will readily scale to larger collections.

3.1. Using Keyphind

Keyphind's interface is shown in Fig. 1. A user initiates a query by typing a word or phrase and pressing the "Search" button, just as with other search engines. However, what is returned is not a list of documents, but rather a list of keyphrases containing the query terms. Since all phrases in the database are extracted from the source texts, every returned phrase represents one or more documents in the collection. Searching on the word *text*, for example, returns a list of phrases including *text editor* (a keyphrase for 12 documents), *text compression* (11 documents), and *text retrieval* (10 documents) (see Fig. 1). The phrase list provides a high-level view of the topics covered by the collection, and indicates, by the number of documents, the coverage of each topic.

Following the initial query, a user may choose to refine the search using one of the phrases in the list, or examine one of the topics more closely. Since they are derived from the collection itself, any further search with these phrases is guaranteed to produce results — and furthermore, the user knows exactly how many documents to expect. To examine the documents associated with a phrase, the user
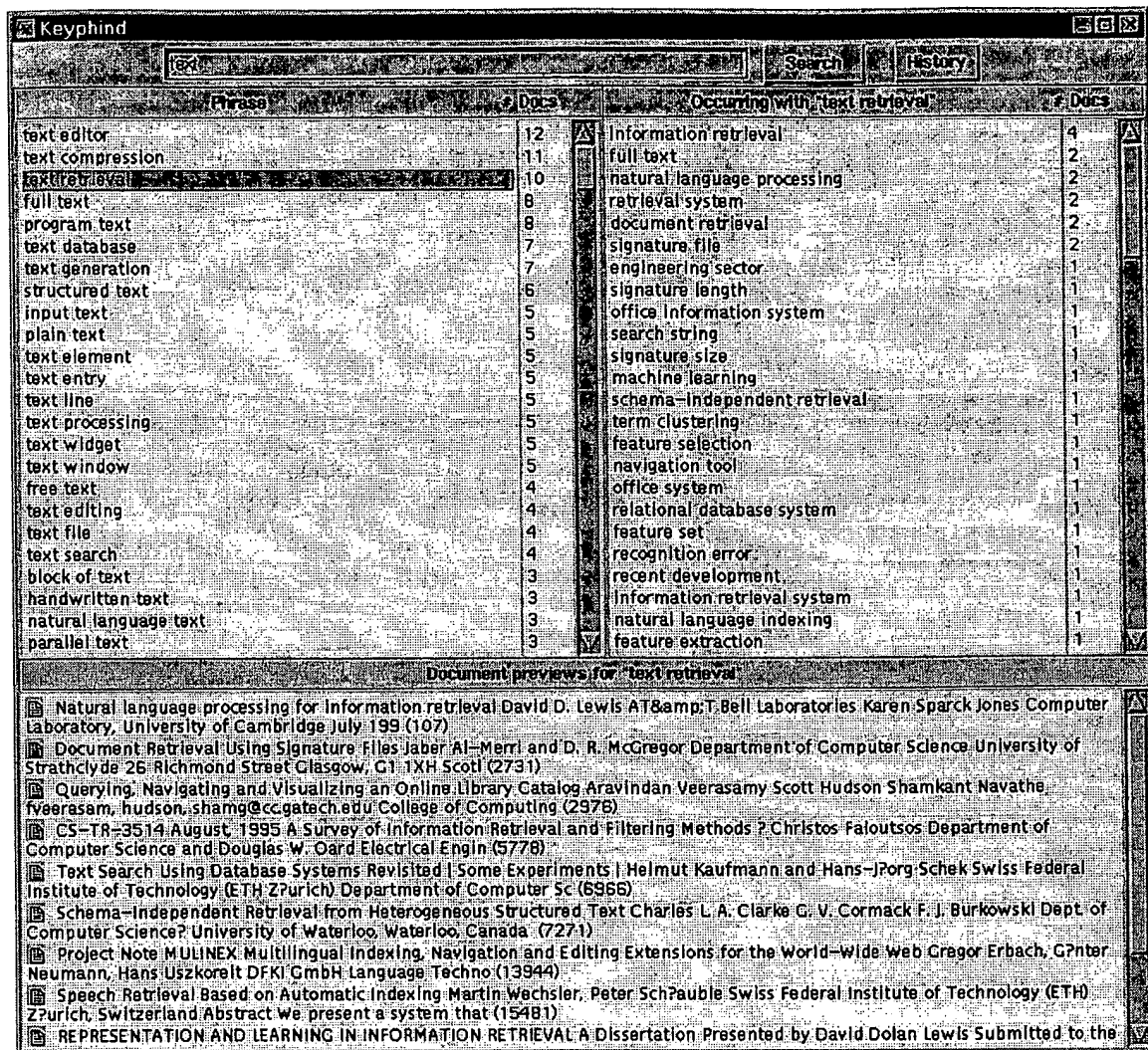
Fig. 1. Keyphind user interface.

selects one from the list, and previews of the documents are displayed in the lower panel of the interface. Selecting any document in the preview list shows the document's full text. More detail on Keyphind's capabilities is given in later sections.

3.2. Automatic keyphrase extraction

Keyphrases are words or short phrases, commonly used by authors of academic papers, that characterize and summarize the topics covered by a document. Keyphrases are most often noun phrases of between two and four words. For example, the keyphrases we have chosen for this article are *digital libraries*, *browsing interfaces*, *text mining*, *keyphrase extraction*, and *machine learning*. The index terms collected in "back-of-the-book" subject indexes also represent a kind of keyphrase. Keyphrases can be a suitable basis for a searchable index, since they represent the contents and subject matter of a document (e.g., Ref. [42]). In most document collections, however, author-specified keyphrases are not available for the documents; and it is laborious to manually determine and enter terms for each document in

a large collection [10]. A keyphrase index, therefore, is usually feasible only if index terms can be determined automatically.

Automatic determination of keyphrases generally works in one of two ways: meaningful phrases can either be *extracted* from the text of the document itself, or may be *assigned* from some other source (but again using the document text as input). The latter technique, keyphrase assignment, attempts to find descriptive phrases from a controlled vocabulary (e.g., MeSH terms) or from a set of terms that have been previously collected from a sample of the collection (e.g., Ref. [37]). Assignment is essentially a document classification problem, where a new document must be placed into one or more of the categories defined by the vocabulary phrases [27,34]. In general, the advantage of keyphrase assignment is that it can find keyphrases that do not appear in the text; the advantage of keyphrase extraction (the technique used in Keyphind) is that it does not require an external phrase source. We now consider keyphrase extraction in more detail.

There are two parts to the problem of extracting keyphrases: first, candidate phrases must be found in the document text, and second, the candidates must be evaluated as to whether they are likely to be good keyphrases. A variety of techniques have been proposed for each of these steps, as described below.

3.2.1. Finding candidate phrases

Since any document will contain many more multi-word sequences than keyphrases, the first step in extracting keyphrases is finding reasonable candidate phrases for further analysis. There are two main approaches to finding phrases in text: syntactic analysis and statistical analysis [15,41]. Syntactic analysis parses the text and reports phrases that match predetermined linguistic categories. Often, these categories define noun-phrase forms that can be determined using a part-of-speech tagger (e.g., Refs. [2,40]). More sophisticated analysis is also possible: for example, Srtzalkowski et al. [42] look for head-modifier pairs, where the head is a central verb or noun, and the modifier is an adjunct argument of the head.

In contrast, finding phrases through statistical analysis does not consider the words' parts of speech. Instead, this approach calculates co-occurrence statis-

tics for sequences of words; those words that appear frequently together are likely to indicate a semantically meaningful concept [41]. This technique generally uses a stop-word list to avoid finding phrases made up of function words (e.g., *of*, *the*, *and*, *then*). In its simplest form, statistical analysis simply finds word sequences that occur more frequently than some fixed threshold [41].

3.2.2. Evaluating candidate phrases as keyphrases

The second step in finding keyphrases involves determining whether a candidate word sequence is or is not a keyphrase. This process generally means ranking the candidates based on co-occurrence criteria and then selecting the top-ranked phrases. The techniques used are similar to those for term weighting and feature selection in vector-space models, where a few components of a multi-dimensional term vector are chosen to represent the document [18,26]. Two common criteria for ranking candidates are simple frequency (normalized by the length of the document), and TF × IDF measures, which compare the frequency of a phrase in the document with its rarity in general use. Other possible criteria include entropy measures [34], first occurrence in the document (see below), or appearance of the phrase in "important" sections of structured text (e.g., title tags in HTML documents). Once phrases are given a score based on these criteria, the best keyphrases are those candidates with the highest ranks.

However, one problem of evaluating candidates is that when several attributes contribute to a phrase's score, it is difficult to construct a simple model for combining the variables. Machine learning techniques can be used to build complex decision models that can take several attributes into account, and this is the approach used by Keyphind. Machine learning has been used for a variety of purposes in retrieval systems (e.g., Refs. [8,34]), but only one other project to our knowledge uses it for ranking keyphrases. This is the Extractor system [46], which uses a genetic algorithm to determine optimal variable weights from a set of training documents. In the current project, we use a technique that is faster and simpler than genetic algorithms, but provides equal performance. Our extraction process is described below.
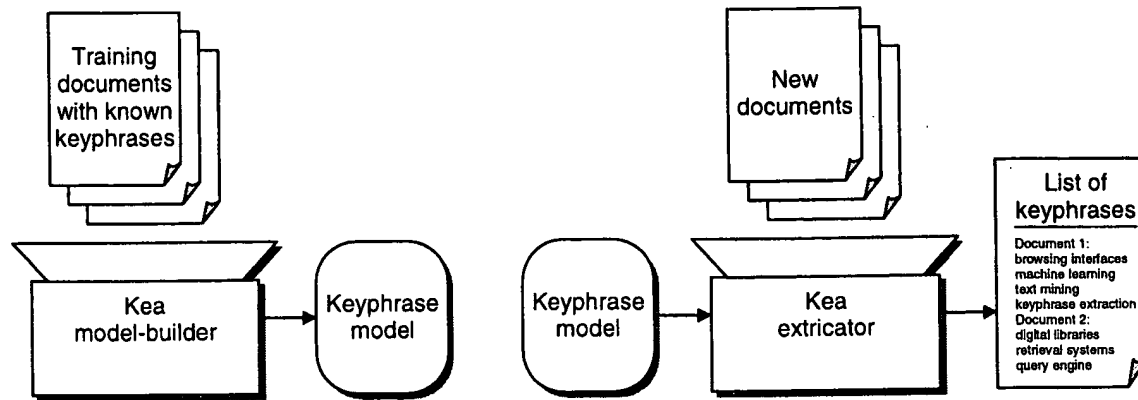
Fig. 2. Kea keyphrase extraction process.

3.3. Keyphrase extraction in keyphind

We have previously developed a system called Kea [16] to extract keyphrases from text. [6] It uses machine learning techniques and a set of training documents to build a model of where keyphrases appear in documents and what their distinguishing characteristics are. In particular, we treat extraction as a problem of supervised learning from examples, where the examples are documents in which every phrase has been manually classified as a keyphrase or nonkeyphrase. This manual classification of phrases is accomplished using the set of author-specified keyphrases for the document. The learning process results in a classification model, and this model is used to find keyphrases in new documents.

Like other approaches, Kea extracts keyphrases using the two steps described above: first identifying candidate phrases in a document's text, and then ranking the candidates based on whether they are likely to be keyphrases. However, we differ from previous work in that we rank candidates using a model built with the Naïve Bayes machine learning algorithm [25]. The steps involved in training and extraction are illustrated in Fig. 2, below, we describe how Kea finds and ranks phrases, and then discuss the construction of the Naïve Bayes model in more detail.

---

[6] Several versions of Kea have been implemented; the version described in Ref. [16] differs from the present description in that it uses an extensive stopword list rather than a part-of-speech tagger to identify candidate phrases in a document.

3.3.1. Finding phrases

The process used to find candidate phrases for Keyphind is similar to that used in earlier projects (e.g., Refs. [30,40]). First, documents are cleaned and tokenised, and then tagged using the Brill part-of-speech tagger [4]. The tagger adds a part-of-speech indicator (e.g., /NN for noun, /VB for verb) to each word. Words are not currently stemmed, although we do fold plurals to singular forms (e.g., "libraries" to "library") and standardize words that have different spellings (e.g., "labor" to "labour"). Second, all phrases matching certain lexical patterns are reported. In particular, we accept any sequence of words consisting of a string of nouns and adjectives with a final noun or gerund. For example, "probabilistic machine learning" contains an adjective followed by a noun followed by a gerund. This pattern was proposed initially by Turney [46] and is similar to others used in the literature (e.g., Ref. [30]); in our own experiments with author-generated keyphrases, the pattern covered more than 90% of a test set of more than 1800 examples.

3.3.2. Building the Naïve Bayes ranking model

The machine learning technique that we use to build the ranking model requires a set of training documents where positive instances (that is, "true" keyphrases) have already been identified. For our training data, we used papers from the CSTR where the authors have already specified a set of keywords and phrases. Although the author's choices are not always the best examples of good keyphrases (for example, there are often good keyphrases in the text

that were not chosen by the author), this method is simple and readily available.

The process of building the ranking model involves four preliminary steps. First, author keyphrases are removed from the training documents and stored in separate files. Second, candidate phrases are found in the training documents as described above (note that some author keyphrases do not appear in the document text). Third, we calculate values for three attributes of each phrase:

- distance (d) is the distance from the start of the document to the phrase's first appearance (normalized by document length)
- term frequency (tf) is the number of times the phrase appears in the document (normalized by document length)
- inverse document frequency (idf) is the phrase's frequency of use in the domain of the collection. This attribute is approximated using a random sample of 250 documents from the collection.

Term frequency and inverse document frequency are combined in a standard TF × IDF calculation [50]. These attributes were experimentally determined to be the most useful in characterising keyphrases [16]: that is, keyphrases are more likely to appear early in a document, to appear often in the document, and to appear less often in general use. Fourth, each candidate phrase is marked as a keyphrase or a nonkeyphrase, based on the author-specified keyphrases for that document.

Once these preliminary steps are completed, the Naïve Bayes machine-learning algorithm compares attribute values for the known keyphrases with those for all the other candidates, and builds a model for determining whether or not a candidate is a keyphrase. The model predicts whether a candidate is or is not a keyphrase, using the values of the other features. The Naïve Bayes scheme learns two sets of numeric weights from the attribute values, one set applying to positive ("is a keyphrase") examples and the other to negative ("is not a keyphrase") instances.

3.3.3. Ranking candidate phrases using the Naïve Bayes model

To rank each candidate phrase that has been found in a new document (one where there are no author keyphrases), Kea first calculates values for the three attributes described above. The Naïve Bayes model built in the training phase is then used to determine a probability score, as follows. When the model is used on a candidate phrase with feature values t (TF × IDF) and d (distance of first occurrence), two quantities are computed:

$$P[t,d,\text{yes}] = P_{\text{TF} * \text{IDF}}[t|\text{yes}] * P_{\text{distance}}[d|\text{yes}] * P[\text{yes}] \quad (1)$$

and a similar expression for $P[t,d,\text{no}]$, where $P_{\text{TF} * \text{IDF}}[t|\text{yes}]$ is the proportion of phrases with value t among all keyphrases, $P_{\text{distance}}[d|\text{yes}]$ is the proportion of phrases with value d among all keyphrases, and $P[\text{yes}]$ is the proportion of keyphrases among all phrases (i.e., the prior probability that a candidate will be a keyphrase). The overall probability that the candidate phrase is a keyphrase can then be calculated:

$$p = P[\text{yes}] / (P[\text{yes}] + P[\text{no}]). \quad (2)$$

Candidate phrases are ranked according to this value. When all of the document's phrases are ranked, the top 12 candidates are selected and written to a file for later indexing.

Further details of the extraction process and the Naïve Bayes scheme can be found in other articles (e.g., Ref. [16]).

3.3.4. Performance

We have evaluated the Kea algorithm in terms of the number of author-specified keyphrases that it correctly extracts from text. In our experiments, [7] Kea's recall and precision were both approximately 0.23 — therefore, it finds about one in four author-specified keyphrases. We also examined the question of how many keyphrases to extract, and considered both a rank cutoff and a probability cutoff. The experiments showed that selecting about 12 candidates usually includes all of the high-probability phrases (where "high" is empirically determined), without including too many poor phrases.

For this research, we have examined Kea's output for about 20 CSTR documents. We estimate that an average of 9 of the 12 keyphrases extracted for each

---

[7] In these experiments, we used a combined measure of recall and precision called the F-measure.

document could be considered useful to an end-user (some of these were ungrammatical, but still understandable), and about 3 of 12 were unusable. An example of the choices, using this article as input, is shown in Table 2.

The performance of our algorithm represents the current state of the art for machine-learning approaches (see Ref. [16]); however, recall and precision are still quite low. There are several reasons for this. First, only about 80% of the author's keyphrases actually appear in the text of the document, and this forms an upper bound on the number of phrases that can be extracted by any algorithm. Second, attempting to match author phrases complicates the machine-learning problem because the data is hugely skewed towards negative instances. For every author-specified keyphrase in a document of the CSTR, there are about 1000 phrases that are not keyphrases. Third, extracting author keyphrases is a more demanding task than simply extracting "good" keyphrases (although it is much easier to evaluate); even two human indexers will rarely come up with exactly the same index terms for a document [17].

The primary problem of evaluating our technique using author-specified keyphrases is that the scheme does not assess the quality of those keyphrases that do *not* match author phrases. We make the assumption that maximizing performance on extraction of author keyphrases will also maximize the quality of all extracted phrases. However, we are currently investigating other assessment techniques: in particu-

lar, we are constructing a corpus of documents where human indexers have marked (and ranked) *all* phrases in the text that could be used as valid keyphrases.

Kea processed the collection of 26 432 documents in about 4 days (on a Pentium 233-MHz system running Linux); more than half of this time was used by the part-of-speech tagger. The resulting raw keyphrase file contains 309 908 keyphrases, about 6 Mb of text. More than half of the raw phrases are duplicates from some other document; as described below, the list of unique phrases contains only 145 788 entries.

3.4. Indexing

Once keyphrases are extracted from each document in the collection, Keyphind converts the raw keyphrase file into indexes that are used by the query interface. This process, illustrated in Fig. 3, produces four indexes: a phrase list, a word-to-phrase index, a phrase-to-document index, and a document-to-phrase index. The phrase list simply associates each unique phrase with a number that is used in the remaining indexes (for efficiency reasons); the other indexes are described below.

The *word-to-phrase index* lists all words in the document collection and indicates for each one the phrases that contain it. For example, "browsing" appears in "browsing technique," "collaborative browsing," "browsing interface," etc. This is the only index needed to process basic queries: the sets of phrases for each word in the query are found and then intersected to find the phrases containing *all* of the query terms.

The *phrase-to-document index* lists every phrase in the database and indicates which documents they were extracted from. Documents are represented by numbers, so the entry for "browsing technique" might show document numbers 113, 4024, 75 376, and 165 234. This index is used to retrieve the appropriate documents when a phrase is selected, and to calculate the co-occurring phrases for any selected phrase (see below).

The *document-to-phrase index* lists every document by number and indicates all phrases that were extracted from that document. There are 12 phrases for each document. The document-to-phrase index is

Table 2
Example output from Kea

| Phrases chosen by the authors	Phrases chosen by Kea
Browsing interfaces	1. Digital libraries
Digital libraries	2. Browsing in digital libraries
Text mining	3. Keyphrase index
Keyphrase extraction	4. Libraries with keyphrase
Machine learning	5. Digital libraries with keyphrase
	6. Browsing tasks
	7. Keyphrase extraction
	8. Usability study
	9. Browsing interfaces
	10. Search engines
	11. Document collections
	12. Query engines
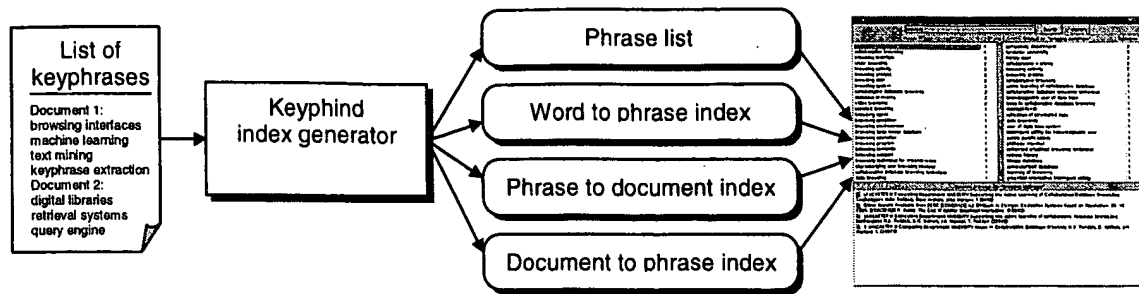
Fig. 3. Generation of Keyphind indexes.

used only in the calculation of co-occurring phrases (see below).

Keyphind's indexes are small, and quick to create. It took less than 5 min to build the indexes for the 26 000-document example collection (on a Pentium Pro 200-MHz system running Linux); this speed, however, is dependent on being able to hold the entire phrase list in a hash table. More important than creation time, however, is index size, and the indexes are tiny in comparison to the source text. The file sizes for the four indexes created for the example collection are shown in Table 3. Since these indexes are derived from 1 Gb of source text, the keyphrase approach offers an index that is less than 1% of the size of the source. In comparison, standard full-text inverted indexes are typically larger than the source text, and even modern compressed indexes (e.g., Ref. [50]) are about one-tenth the size of the source. Of course, the small size of the indexes is only achieved because we discard all but 12 phrases for each document — but we believe that for some browsing tasks, indexing 12 meaningful phrases may be as good as indexing every word.

Keyphind stores all of these indexes in memory. The word-to-phrase index is stored as a hash table for quick look-up, and the other three are stored as sequential arrays. The memory space needed is about 20 Mb. The small size of these indexes, both in memory and on disk, means that keyphrase indexes can be downloaded to a networked client at the start of a browsing session, providing quick access and short response times.

3.5. Presentation and user interaction

Keyphind's query interface is shown in Figs. 1 and 4. There are three activities that users can undertake. They can issue keyphrase queries, they can view and preview documents associated with a keyphrase, and they can work with co-occurring phrases related to their current selection. We describe each of these below.

3.5.1. Keyphrase queries

Being based on keyphrases rather than documents, Keyphind answers a slightly different question than does a full-text search engine. Instead of deciding which *documents* contain the user's query terms, Keyphind determines which *keyphrases* contain the query terms. When the user enters a word or topic and presses the "Search" button, the system consults the word-to-phrase index, retrieves those phrases that contain all of the query words, and displays them in the upper-left panel of the interface as the result of the query. In addition, the documents associated with each phrase in the result set are counted (using the phrase-to-document index) and the number is shown alongside the phrase. The result set can be sorted either by phrase or by number of documents. An example of a basic query on *extraction* is illustrated in Fig. 4a.

Table 3
File sizes of Keyphind indexes

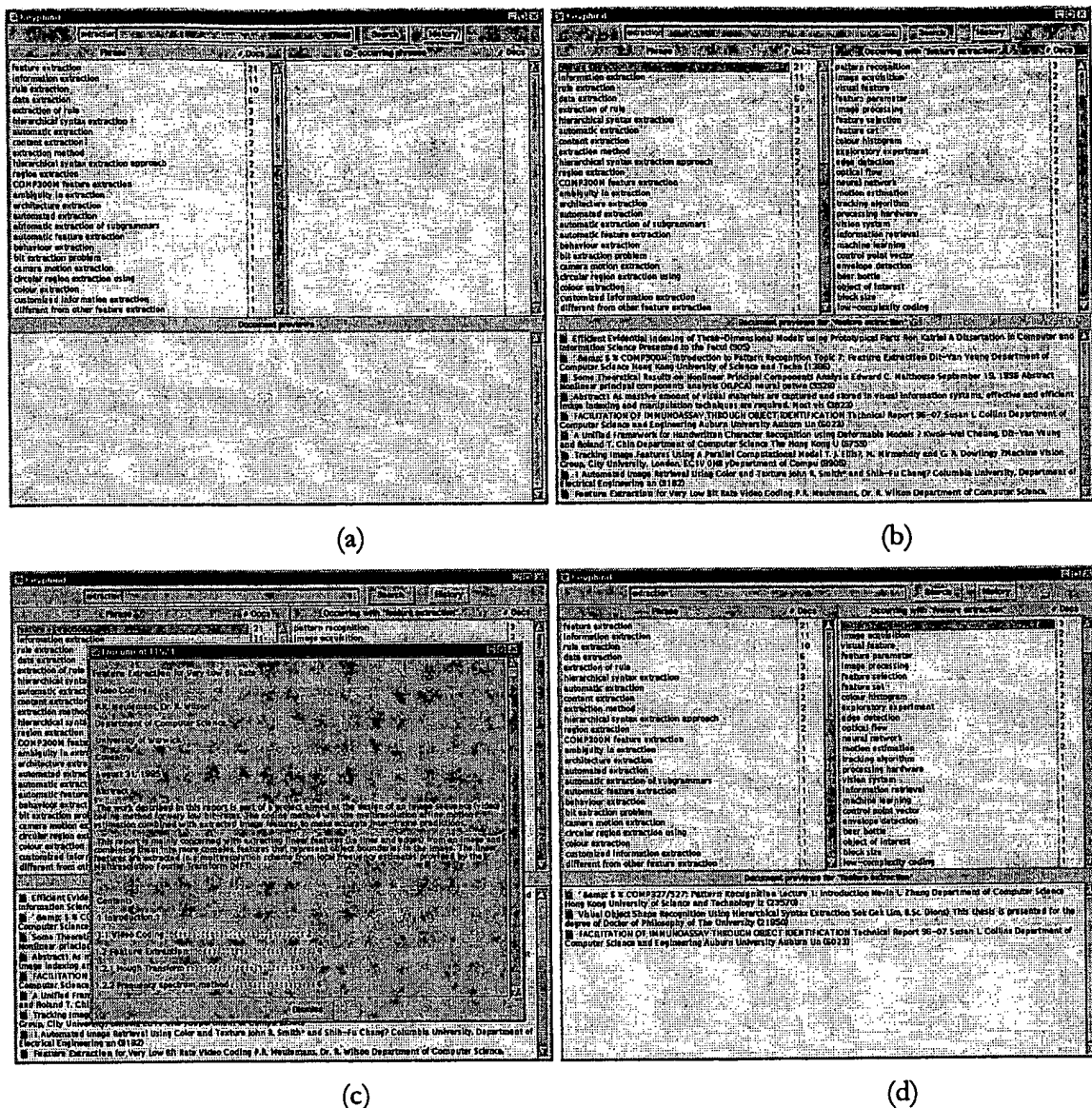| Index	Number of items	File size
Phrase list	145 788 Unique phrases	2.98 Mb
Word-to-phrase index	35 705 Unique words	2.80 Mb
Phrase-to-document index	145 788 Entries	1.84 Mb
Document-to-phrase index	26 432 Documents	1.79 Mb
Total		9.41 Mb

Fig. 4. Four types of interaction with Keyphind. (a) Basic query results. (b) Document previews and co-occurring phrases. (c) Document inspection window. (d) Co-occurrence filtering of previews.

3.5.2. Refining the query

The result set contains all keyphrases in the database that contain the query terms. Therefore, this list shows all possible ways that the user's query can be extended with additional words. Additional queries can be issued by double-clicking on phrases in the list.

3.5.3. Previewing and viewing documents

Once a keyphrase query has been made, the user can look at the documents associated with any of the phrases displayed. When the user selects one, Keyphind retrieves the first few lines of each associated document and displays these previews at the bottom of the interface (Fig. 4b). If the phrase of

interest occurs in the document preview, it is high-lighted in colored text. To look more closely at a document, clicking on the preview presents the full document text in a secondary window (Fig. 4c).

3.5.4. Related phrases and co-occurrence filtering

An additional capability for exploring the collection is based on the co-occurrence of keyphrases. When the user selects a phrase from the result list, additional phrases related to the user's selection are displayed in the upper-right panel of the interface. These related phrases can provide the user with new query ideas and new directions for exploring a topic. The scheme works as follows.

(1) The user selects a phrase from the result list, such as *feature extraction* (Fig. 4b).

(2) *Feature extraction* is a keyphrase for 21 documents; however, since we originally extracted 12 keyphrases from each text, each of the 21 documents is also associated with 11 other keyphrases. These are called "co-occurring phrases."

(3) Sometimes, a phrase will co-occur with *feature extraction* in more than one of the 21 documents, and Keyphind keeps track of this number.

(4) The system puts the co-occurring phrases into a list and displays them in the top right panel of the interface (Fig. 4b), along with the number of co-occurrences. For example, *pattern recognition* co-occurs with *feature extraction* in three documents, and *image acquisition* co-occurs twice.

The co-occurrence number is exactly the number of documents that would be returned if the user formed a query from both the target keyphrase AND the co-occurring phrase. Thus, related phrases can be used to filter the documents in the previews list. When the user selects a phrase in the co-occurrence list, the lower panel of the interface shows only those documents containing both keyphrases. This situation is illustrated in Fig. 4d, where the user has selected *feature extraction* and then clicked on *pattern recognition* in the co-occurrence list; the documents shown in the preview panel are the two that contain both *feature extraction* and *pattern recognition* as keyphrases.

3.6. Summary

By using keyphrases as the basic unit of indexing and presentation, Keyphind allows users to interact with a collection at a higher level than individual documents. This approach supports browsing tasks in four ways.

(1) *Topical orientation.* Queries return a list of keyphrases rather than documents, giving users an immediate idea of the range of topics that the collection covers. This is useful in both evaluating the collection and in exploring a subject area.

(2) *Phrase-based clustering.* Each keyphrase represents several documents, so users can see and work with a much larger portion of the collection than they could with document-based searching. In addition, the reason why documents have been clustered together is easily understood — they share a keyphrase.

(3) *Query refinement.* Since the returned phrases are almost always longer than the search string, the phrase list shows all possible ways that the query can be extended and still return a result. In addition, the co-occurring phrases provide a means for finding terms that are related to the user's query, and can be used to filter the query results, explore new directions, and help address the vocabulary problem (e.g., Ref. [7]).

(4) *Results prediction.* By showing how many documents there are for each phrase, Keyphind indicates the collection's depth and also what will be returned when the query is extended.

4. Evaluation: a usability study of Keyphind

We hypothesized that using keyphrases as the basis for indexing and clustering documents would support browsing in large collections better than conventional full-text search does. To test that hypothesis, and to determine the strengths and weaknesses of our approach, we conducted a small usability study in which people carried out browsing tasks with Keyphind and also with a traditional query interface. The goals of this study were to better understand how people use phrases in searching, to determine differences in people's search strategies when they used the two interfaces, and to make an initial assessment about whether working with keyphrases makes browsing easier. Tasks involved either evaluating the coverage of a collection in an area, or looking for documents relevant to a particu-

lar topic. From our observations and from people's comparisons, it was clear that search strategies differed considerably, and that most participants found the tasks to be easier with the Keyphind interface. The next sections outline our methods and summarize our results.

4.1. Method

Ten computer science students and researchers from the University of Waikato participated in the study as unpaid volunteers. Five of the participants used search engines a moderate amount (1–10 sessions per week) and five searched more frequently (more than 15 sessions per week). Participants were given a short training session on the two interfaces, where the capabilities of each system were explained and where the participant carried out a practice task. Once the training session was complete, the participant began their tasks.

4.1.1. Tasks

Participants completed three tasks using two query interfaces. Two tasks involved assessing the coverage of a collection in a particular area, and one task involved exploring the collection to find documents relevant to a topic. A participant carried out each task first with one interface (Keyphind or traditional) and then repeated the task with the other interface. Order was counterbalanced, so that half the participants started with each interface.

4.1.1.1. Coverage evaluation tasks 1 and 2. Participants were given a task scenario that was intentionally underspecified, asking them to find subtopics of a topic area. For the first task, it was:

"You are a researcher who has been asked to find on-line resources for doing computer science research in *planning*. As one part of your evaluation, please determine three kinds of planning that are adequately represented in the current collection. Adequate representation means that there should be at least three documents that have some relation to the type of planning you are investigating."

For the second task, the topic area was *computer graphics*, and the instructions were otherwise the

same. None of the participants considered themselves to be experts in planning research or computer graphics, so they did not begin either task with a clear idea of what they were seeking. This task was designed to determine how easily people could reach a high-level assessment of a collection.

4.1.1.2. Exploration task. Participants were asked to name a subject area from their current research or studies. They were then asked to find two articles in the collection that were relevant to that topic. This task was designed to find out how people explored an area, and to determine how easily they could find something of use to their work.

4.1.2. Data collection

One experimenter recorded observations about strategy use during the tasks. To assist observation, participants were asked to "think aloud" as they carried out the task. After each task, participants were asked to compare their experiences with the two interfaces (see questions 1–3 of Table 4). We did not ask these questions after the exploration task, since that task was recognized to be highly variable. At the end of the session, the experimenter conducted a short interview. Four specific questions were asked (see questions 4–7 of Table 4), and then the experimenter asked additional questions to explore particular incidents observed during the tasks.

4.1.3. Systems

Keyphind is illustrated above in Figs. 1 and 4; the traditional interface that we compared it to is shown

Table 4
Questions asked after tasks and during interview

Questions asked after tasks:
1. Was it easier to carry out the task with one or the other of these systems?
2. If yes, which one?
3. If yes, was the task: slightly easier, somewhat easier, or much easier?

Interview questions:
4. What process did you use to find out about a topic with each system?
5. What were the major differences between the two systems?
6. Would you use a system like this one (Keyphind) in your work?
7. How would you use it?

in Fig. 5. This system is the standard WWW interface used with the NZDL (www.nzdl.org). Participants were allowed to change the search parameters in this interface (turning word stemming on or off, folding case, using boolean or ranked retrieval, phrase searching) as they wished. Search parameters in Keyphind, however, were fixed as described above.

4.2. Results

All but two participants were able to complete the tasks in both interfaces. In these two cases, technical

problems prevented them from using the traditional interface. Our results are organized around two issues: first, which interface (if any) made the task easier; and second, how search strategies varied with the two interfaces.

4.2.1. Which interface made the tasks easier

We asked whether the first two tasks were easier with either interface, and if so, how much easier. We did not do a formal comparison for the third task, due to its open-ended nature. Participants' responses are shown in Table 5. Two participants were unable
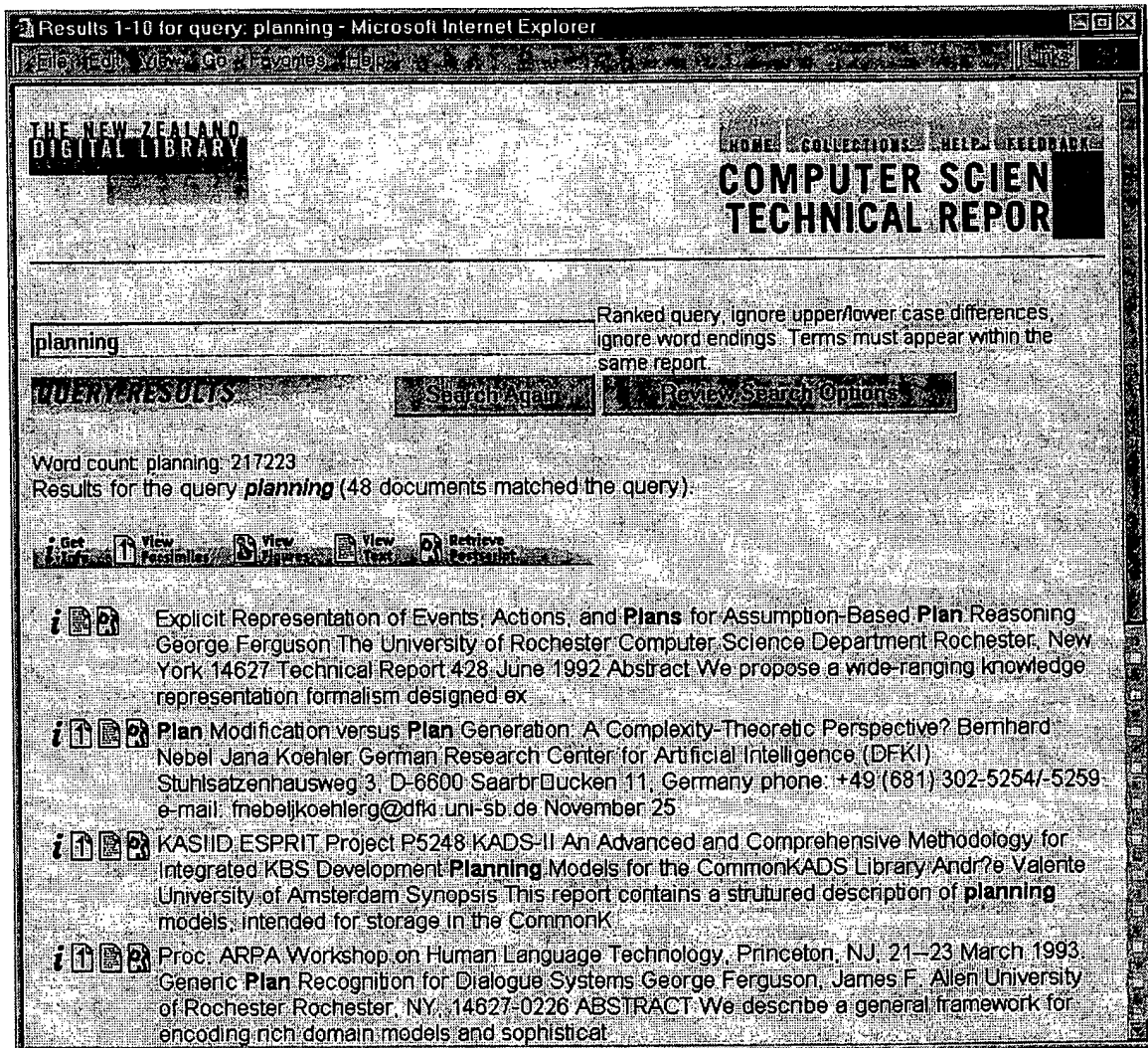

Fig. 5. Traditional WWW query interface.

Table 5
Responses to comparison questions (cells indicate number of people)

Task	The task was easier with:			
	Keyphind	Traditional	No difference	No answer
1 (Planning)	7	0	1	0
2 (Computer graphics)	3	0	4	1

Task	How much easier:			
	Slightly	Moderately	Much	No answer
1 (Planning)	2	0	4	1
2 (Computer graphics)	1	0	2	0

to compare the interfaces because of technical problems with the traditional interface.

Seven of eight people found the first task to be easier using the Keyphind interface, and three found the second task easier as well. No one found the tasks easier with the full-text interface, although several participants felt that there was no difference. When we considered the responses in terms of a participant's stated level of search-engine use, we did not find any clear distinctions between the moderate users (less than 10 searches per week) and the more frequent users (more than 15 searches per week).

4.2.2. How the interfaces affected strategy use

Several differences were observed in people's strategies, differences that appear to relate to the way that the two interfaces present information. Most people began the tasks in the same way — by entering a general query and then looking through the results — but from there, the two groups diverged. In the full-text interface, people looked for information by scanning the document previews returned from the general query; in Keyphind's interface, they looked through the list of returned phrases. For example, the first task asked people to find types of planning covered by the collection. Several participants reported that in the full-text interface, they looked for phrases containing "planning" and kept track in their heads of how often they saw each phrase. Although "planning" was emphasized in the previews, this strategy could be laborious, since the previews were not formatted and showed only the

first 200 characters of the file. In contrast, all the phrases in Keyphind's result list contained *planning*, and so people moved immediately to investigating those phrases that looked reasonable and were associated with several documents. The initial results returned for both interfaces are shown in Fig. 6.

In the second collection–evaluation task, with the topic area *computer graphics*, strategies with the full-text interface were much the same as in the first. In Keyphind, however, the results of the initial query did not contain many phrases that were clearly subtopics of computer graphics, and the strategy of simply looking through the phrase list did not work as well. In addition, most of the documents were concentrated in one phrase — *computer graphics* itself (see Fig. 7). Some participants dealt with this situation by showing the previews for this phrase, and then scanning the previews much as they did in the full-text interface. Four people, however, used the co-occurring phrases panel to look for possible subtopics of computer graphics; and there were several reasonable phrases, such as *ray tracing, volume rendering,* and *virtual reality* (see Fig. 8).

The final task asked people to find two technical reports relevant to their area of research or study, and there were fewer strategy differences in this task. Participants began in the same way that they began the other tasks — by entering the research area as a query. In the Keyphind interface, people spent more time looking through document previews than in the other tasks, probably because the final task involved looking for documents rather than topics. Most participants used the following strategy: enter the re-
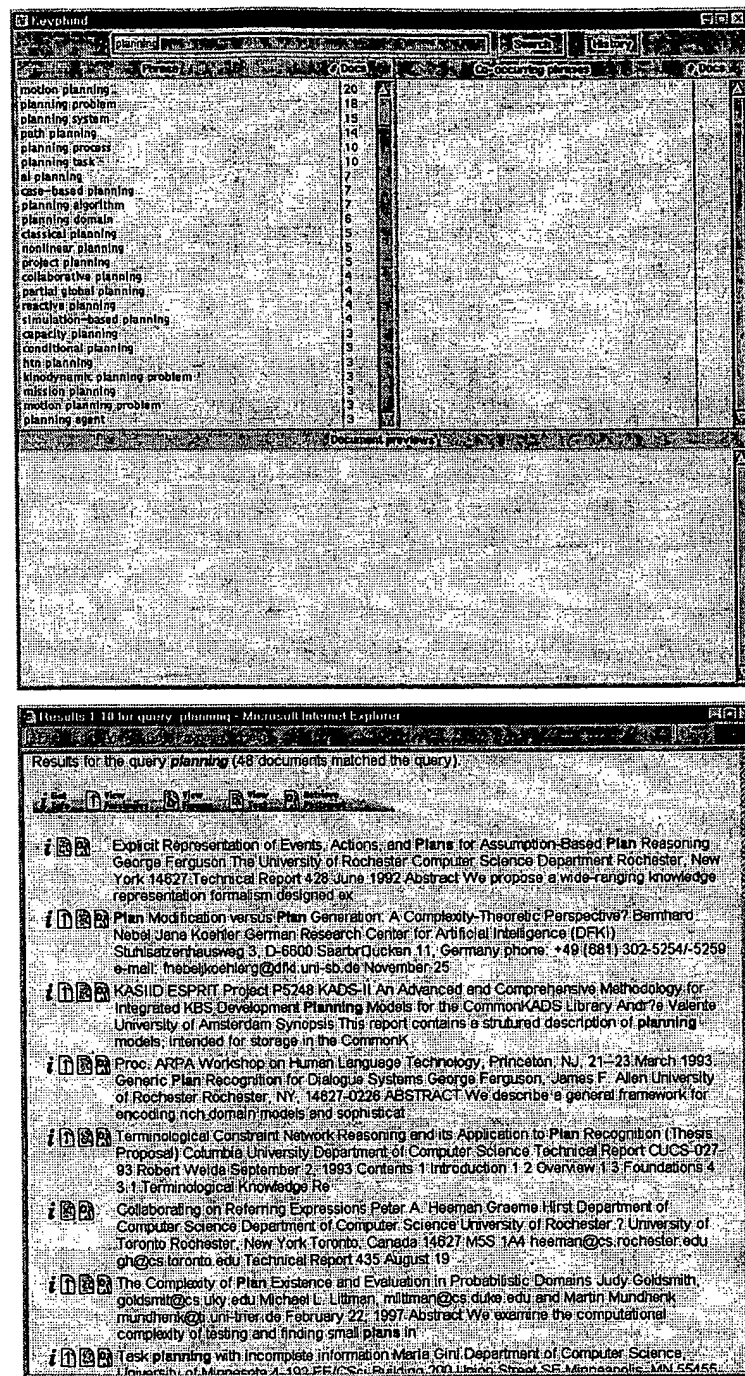
Fig. 6. Results from the query *planning* in Keyphind (above) and traditional interface.

search area as the initial query, choose a likely looking phrase from the results list, and then scan through the document previews for that phrase.

With the full-text interface, people once again carried out the task by looking through the document previews returned from the query. In this task, how-
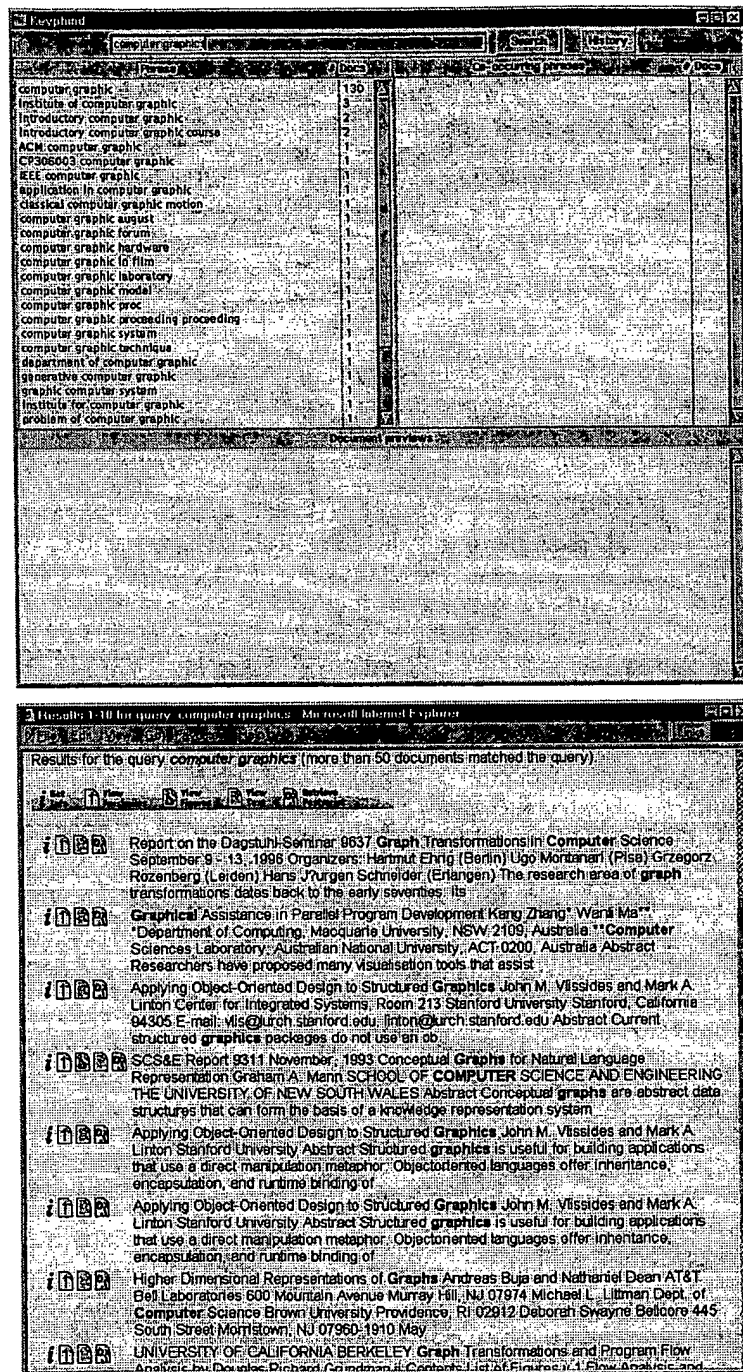
Fig. 7. Results from the query *computer graphics* in Keyphind (above) and traditional interface (below).

ever, people were already familiar with the area, and so they already knew what subtopics they were looking for. That meant that when they did not find what they wanted, they (in most cases) knew of other

Fig. 8. Most common phrases co-occurring with *computer graphics* in Keyphind (see upper right panel).

search terms to try. Although people were on the whole successful in this task, there was some difficulty finding query terms that were appropriate to the ranked retrieval algorithm. For example, one participant entered *computer* and *vision* as query terms and was given a large set of documents where *computer* figured prominently, but that had nothing to do with computer vision. In most cases, changing the system's settings to do a string search for the phrase led to more relevant results.

4.2.3. Other interview results

At the end of the session, we asked participants about the differences they saw between the two systems, and about how they might use a phrase-based system like Keyphind in the queries they performed for their research. The major differences seen by almost all of the participants were that Keyphind presented topics rather than documents, and that it was more difficult to explore an area by scanning through documents than by looking through a list of topics. However, three participants said that while Keyphind provided a faster and easier route to the subtopics of an area, they were less confident

that the list of subtopics was complete than they were when they (laboriously) found subtopics by scanning document previews in the traditional interface. In addition, three participants said that they particularly liked being able to see how many documents would be returned from a further search.

When asked about how a phrase-based system could be used in their day-to-day searches, seven of the participants said that they would like to have a system like Keyphind available in addition to their normal search tools (although not as a replacement). Participants said that they would use such a system in situations where they were having difficulty with their traditional tools, or to "familiarize themselves with [an] area" when they first started to explore.

5. Discussion

The usability study provides evidence that a phrase-based query system can provide better support than traditional interfaces for browsing tasks, using an index that is far smaller than a full-text index. However, our experiences with Keyphind have

suggested several issues in the keyphrase approach that must be considered in more detail. In this section, we look at two main areas: first, limitations caused by the structure and format of a keyphrase approach, and second, issues involved with phrase quality and how users react to an imperfect index.

5.1. User queries and automatically extracted keyphrases

Information retrieval literature suggests that using phrases assists some tasks and some queries, but does not greatly affect others (e.g., Refs. [30,32]). This seems to be the case with Keyphind as well, and the problem appears to lie in the fact that some kinds of user queries better fit the structure of the automatically extracted phrase hierarchy. Borman [3] notes that one problem in traditional query systems is that successful searching requires that users know about the underlying IR techniques; and although we believe the keyphrase approach provides a much more transparent model, it does still require that users "think in keyphrases" to a certain extent.

For example, Keyphind was seen as more useful in the first task of the usability study; one likely reason for this is that people's initial queries in the first task returned a broader range of topics. An initial query on *planning* results in several phrases that can be considered subtopics of planning (e.g., *motion planning, case-based planning, collaborative planning*); however, *computer graphics* does not produce such obvious good subtopics. In short, this difference arises because many useful phrases fit the pattern ⟨*something*⟩ *planning*, but fewer fit ⟨*something*⟩ *computer graphics*. This situation is caused in part by the shallowness of the phrase hierarchy, which reduces the richness of the clusters when using a two-word phrase as a query. Note, however, that the case of *computer graphics* is exactly the kind of situation where the co-occurring phrases can provide an alternate set of clusters.

A second problem in matching user queries to the index is the vocabulary problem. Keyphind works best when initial queries are general and short, such as *planning*; the user can peruse the range of possible extensions to the general query and choose those that are most applicable. If users begin with a more specific query, however, they may choose the

'wrong' phrase and so miss related topics. For example, *path planning* returns only seven phrases, even though there are many more phrases under the closely related *motion planning*. This problem could be addressed in part by using a thesaurus to automatically or interactively expand searches (although this would add another index) (e.g., Ref. [9]).

Finally, some of the queries that people make to systems like Keyphind are legitimate topics, but are unlikely to appear as phrases. For example, there are undoubtedly reports relevant to *evaluation of browsing interfaces* in a 26 000-paper collection of CSTR — but this phrase is unlikely to appear in the index because it is unlikely to be extracted as a keyphrase. Keyphind will therefore not return anything for this query. There are several possible ways to reduce the problem, such as automatically performing multiple searches on parts of the query, but the fact remains that keyphrases only approximate a true human-built subject index. Having several search tools that can make up for each others' shortfalls may be the only way to truly solve the problem.

5.2. Phrase quality and completeness

A user of Keyphind is in constant interaction with the phrases that have been automatically extracted from documents in the collection. For a phrase-based approach to be considered useful by its users, people must believe that the system has done a reasonable job of extracting important phrases from documents without including too much garbage. Unfortunately, no fully automatic system can come close to the quality of a human indexer, and so users are going to encounter both missing phrases and poor phrases.

In our view, there are three issues that affect the quality of the phrase index: phrases may be linguistically unusable (e.g., garbage strings), phrases may not be good descriptors of the document they were extracted from, or the "good" keyphrases contained in a document may have been missed by the extractor. Each of these problems reduces recall in the results set; however, in our experience with Keyphind, recall has not been a problem. In general, users seem willing to overlook a certain number of nonrelevant documents (perhaps they have been well trained in this regard by traditional search engines). Second, in large collections, recall may not be as

much of a problem as precision: there are often so many documents that enough relevant items will be returned for the user's purposes, even though many have been missed by the indexing system. This assumes that users will be willing to work around the incorrect results, as long as they can find a reasonable set of documents in a reasonable time.

However, in our experience with the NZDL, we have also found that some frequent users of traditional query systems prefer to carry out browsing tasks by looking over all of the documents and making up their own minds about the structure of the domain, even if this process is more laborious than having an automatically extracted set of phrases. Since Keyphind cannot always provide a perfect set of keyphrases, it is essential to give people easy access to the documents themselves, so that they are not forced to trust the system, but can draw their own conclusions from the raw data.

6. Comparison with previous work

6.1. Phrase-based systems

Phrases have been used in previous information-retrieval systems to improve indexing (e.g., Refs. [15,43]), to help refine queries (e.g., Refs. [1,37]), and to provide facilities for subject browsing (e.g., Refs. [10,28]). In addition, there are a variety of projects that have extracted different kinds of information from text, such as document summaries (e.g., Ref. [24]). In terms of the extraction of keyphrases, Keyphind is most similar to the Extractor system [46], which is fully automatic, and also approaches keyphrase extraction as a supervised learning problem (see Ref. [16] for a comparison with our extraction methods).

Keyphind differs from previous phrase-based systems in that it treats keyphrases as the fundamental unit of indexing and presentation, rather than an add-on to assist one function of a retrieval engine. This means that we use keyphrases as the only index rather than as a supplement to another index, and that we can provide retrieval, clustering, refinement, and results prediction with a single simple mechanism.

6.2. Document clustering

As described earlier, document clustering is often used in browsing interfaces as a means of providing a high-level view of a database. Keyphind differs from other clustering techniques in two ways. First, the mechanism for grouping documents together is the shared keyphrase, which is simple for users to understand — unlike a distance metric. It is easy to see why documents in Keyphind have been clustered together — they all share a common keyphrase. Second, each document in the collection is always part of 12 clusters — one for each extracted keyphrase. This affords a greater degree of overlap than is generally possible with other similarity measures. Since documents may often be about several different themes or topics, it is possible with keyphrase clustering to form an association between parts of a document, rather than demanding that two documents be similar in their entirety in order to be clustered together.

Basing clusters on keyphrases, however, does have some drawbacks not encountered in other methods. First, since documents can fall into more than one cluster, it can be difficult to determine how many documents are actually represented by the phrases returned from a query. Second, the sizes of Keyphind's clusters are variable: a keyphrase may represent hundreds of documents, or only one. Clusters that are too large or too small are less useful to the user; in particular, having many clusters of size 1 is no better than having a list of documents. Therefore, we are considering strategies for splitting large clusters and grouping small ones. For example, we could show some of the phrases within a large cluster (either the extensions of the phrase, or its set of co-occurring phrases); small clusters could be grouped by collapsing component phrases, by folding phrase variants more extensively, or by grouping synonyms based on a thesaurus.

6.3. Projects directly related to Keyphind

Finally, a note on Keyphind's lineage. Keyphind is one of several related projects dealing with the extraction and use of information from text. Kea, discussed earlier, is an ongoing project to extract

keyphrases from text. Keyphind is also descended from an earlier system called Phind [31] that builds phrase hierarchies from the full text of a document. These hierarchies were not based on keyphrases, but on any repeating sequence in the text. The phrase indexes used in Keyphind have also given rise to two other systems: Phrasier [21], a text editor that automatically creates links from phrases in a user's text to documents in a collection, and Kniles (www.nzdl.org/Kea/Kniles), a system that links keyphrases in returned documents to others in the collection.

7. Conclusion

In this article, we described a search engine that supports browsing in large document collections and digital libraries. The Keyphind engine is based on a database of keyphrases that can be automatically extracted from text, and uses keyphrases both as the basic unit of indexing and as a way to organize results in the query interface. Keyphind's indexes are much smaller than standard full-text indexes, are efficient, and are relatively easy to build. Keyphind supports browsing in three main ways: by presenting keyphrases that indicate the range of topics in a collection, by clustering documents to show a larger portion of the collection at once, and by explicitly showing all of the ways a query can be extended. A usability study of Keyphind gives evidence that the phrase-based approach provides better support than full-text engines for some kinds of browsing tasks — evaluation of collections in particular.

8. Software availability

Keyphind is available from www.cs.usask.ca/faculty/gutwin/Keyphind/. Kea, the system used to extract keyphrases for Keyphind, is available from www.nzdl.org/Kea/.
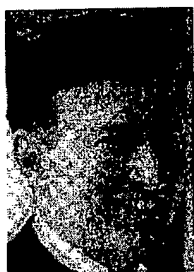
Acknowledgements

References

[1] P. Anick, S. Vaithyanathan, Exploiting Clustering and Phrases for Context-Based Information Retrieval, Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1997, pp. 314–323.

[2] N.A. Bennett, Q. He, C. Chang, B.R. Schatz, Concept extraction in the interspace prototype, Technical Report, Digital Library Initiative Project, University of Illinois at Urbana-Champaign. Currently available from http://www.canis.uiuc.edu/interspace/technical/canis-report-0001.html.

[3] C.L. Borman, Why are online catalogs still hard to use?, Journal of the American Society for Information Science 47 (7) (1996) 493–503.

[4] E. Brill, Some Advances in Rule-Based Part of Speech Tagging, Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI Press, 1994.

[5] S.J. Chang, R.E. Rice, Browsing: a multidimensional framework, in: M.E. Williams (Ed.), Annual Review of Information Science and Technology (ARIST), Vol. 28, 1993, pp. 321–276.

[6] H. Chen, A.L. Houston, Internet browsing and searching: user evaluations of category map and concept space techniques, Journal of the American Society for Information Science 49 (7) (1998) 582–603.

[7] H. Chen, T.D. Ng, A concept space approach to addressing the vocabulary problem in scientific information retrieval: an experiment on the worm community system, Journal of the American Society for Information Science 48 (1) (1997) 17–31.

[8] H. Chen, G. Shankaranarayanan, L. She, A machine learning approach to inductive query by examples: an experiment using relevance feedback, ID3, genetic algorithms, and simulated annealing, Journal of the American Society for Information Science 49 (8) (1998) 693–705.

[9] H. Chen, J. Martinez, A. Kirchhoff, T.D. Ng, B.R. Schatz, Alleviating search uncertainty through concept associations: automatic indexing, co-occurrence analysis, and parallel computing, Journal of the American Society for Information Science 49 (3) (1998) 206–216.

[10] Y. Chung, W.M. Pottenger, B.R. Schatz, Automatic Subject Indexing Using an Associative Neural Network, Proceedings of the 3rd ACM International Conference on Digital Libraries (DL '98), ACM Press, 1998, pp. 59–68.

[11] D.R. Cutting, D.R. Karger, J.O. Pedersen, J.W. Tukey, Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections, Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1992, pp. 318–329.

[12] S. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, R.A. Harshman, Indexing by latent semantic analysis, Journal of the American Society for Information Science 41 (6) (1990) 391–407.

[13] W.M. Detmer, E.H. Shortliffe, Using the internet to improve knowledge diffusion in medicine, Communications of the ACM 40 (8) (1997) 101–108.

[14] K. Doan, C. Plaisant, B. Shneiderman, T. Bruns, Query Previews for Networked Information Systems: a Case Study with NASA Environmental Data, SIGMOD Record, 26, No. 1, 1997.

[15] J. Fagan, Automatic Phrase Indexing for Document Retrieval: An Examination of Syntactic and Non-Syntactic Methods, Proceedings of the Tenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1987, pp. 91–101.

[16] E. Frank, G.W. Paynter, I.H. Witten, C. Gutwin, C.G. Nevill-Manning, Domain-Specific Keyphrase Extraction, to appear in: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1999. Also available from: http://www.cs.waikato.ac.nz/~eibe/pubs/Z507.ps.gz.

[17] G.W. Furnas, T.K. Landauer, L.M. Gomez, S.T. Dumais, The vocabulary problem in human–system communication, Communications of the ACM 30 (11) (1987) 964–971.

[18] D. Harman, Automatic indexing, in: R. Fidel, T. Hahn, E.M. Rasmussen, P.J. Smith (Eds.), Challenges in Indexing Electronic Text and Images, ASIS Press, 1994, pp. 247–264.

[19] M.A. Hearst, C. Karadi, Cat-a-Cone: An Interactive Interface for Specifying Searches and Viewing Retrieval Results Using a Large Category Hierarchy, Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1997, pp. 246–255.

[20] S. Jones, Graphical Query Specification and Dynamic Result Previews for a Digital Library, Proceedings of the ACM Conference on User Interface Software and Technology (UIST '98), ACM Press, 1998, pp. 143–151.

[21] S. Jones, Link as you Type: Using Keyphrases for Automated Dynamic Link Generation, Department of Computer Science Working Paper 98/16, University of Waikato, New Zealand, August 1998.

[22] E. Kandogan, B. Shneiderman, Elastic Windows: A Hierarchical Multi-Window World-Wide Web Browser, Proceedings of the ACM Conference on User Interface Software and Technology (UIST '97), ACM Press, 1997, pp. 169–177.

[23] S. Kirsch, The Future of Internet Search: Infoseek's Experiences Searching the Internet, SIGIR Forum, 32, No. 2, 1998.

[24] J. Kupiec, J. Pedersen, F. Chen, A Trainable Document Summarizer, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1995, pp. 68–73.

[25] P. Langley, W. Iba, K. Thompson, An Analysis of Bayesian Classifiers, Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press, 1992, pp. 223–228.

[26] A. Leouski, W.B. Croft, An evaluation of techniques for clustering search results, Technical Report IR-76, Center for Intelligent Information Retrieval, University of Massachusetts, 1996.

[27] C.H. Leung, W.K. Kan, A statistical learning approach to automatic indexing of controlled index terms, Journal of the American Society for Information Science 48 (3) (1997) 9.

[28] X. Lin, Graphical Table of Contents, Proceedings of the 1st ACM International Conference on Digital Libraries (DL '96), ACM Press, 1996, pp. 45–53.

[29] G. Marchionini, Information Seeking in Electronic Environments, Cambridge Univ. Press, 1995.

[30] M. Mitra, C. Buckley, A. Singhal, C. Cardie, An Analysis of Statistical and Syntactic Phrases, Proceedings of the 5th RIAO Conference on Computer-Assisted Information Searching on the Internet, sponsored by the Centre De Hautes Etudes Internationales D'informatique Documentaire (CID), 1997.

[31] C. Nevill-Manning, I.H. Witten, G. Paynter, Browsing in Digital Libraries: a Phrase-Based Approach, Proceedings of the 2nd ACM International Conference on Digital Libraries (DL '97), ACM Press, 1997, pp. 230–236.

[32] R. Papka, J. Allan, Document Classification using Multiword Features, Proceedings of the Seventh International Conference on Information and Knowledge Management (CIKM '98), ACM Press, 1998.

[33] P. Pirolli, P. Schank, M. Hearst, C. Diehl, Scatter/Gather Browsing Communicates the Topic Structure of a Very Large Text Collection, Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems, Vol. 1, ACM Press, 1996, pp. 213–220.

[34] M. Sahami, S. Yusufali, M.Q.W. Baldonado, SONIA: A Service for Organizing Networked Information Autonomously, Proceedings of the 3rd ACM International Conference on Digital Libraries (DL '98), ACM Press, 1998, pp. 200–209.

[35] G. Salton, Automatic Text Processing, Addison-Wesley, 1989.

[36] G. Salton, J. Allan, C. Buckley, Automatic structuring and retrieval of large text files, Communications of the ACM 37 (2) (1994) 97–108.

[37] B.R. Schatz, E.H. Johnson, P.A. Cochrane, H. Chen, Interactive Term Suggestion for Users of Digital Libraries: Using Subject Thesauri and Co-Occurrence Lists for Information Retrieval Information Retrieval, Proceedings of the 1st ACM International Conference on Digital Libraries, ACM Press, 1996, pp. 126–133.

[38] B.R. Schatz, Information Analysis in the Net: The Interspace of the 21st Century, White Paper for 'America in the Age of Information: A Forum on Federal Information and Communications R&D', National Library of Medicine, U.S. Committee on Information and Communications, 1995.

[39] B. Shneiderman, D. Byrd, W.B. Croft, Sorting out searching: a user interface framework for text searches, Communications of the ACM 41 (4) (1998) 65–98.

[40] A.F. Smeaton, F. Kelledy, User-Chosen Phrases in Interactive Query Formulation for Information Retrieval, Proceed-

ings of The 20th BCS Colloquium on Information Retrieval (IRSG '98), Springer-Verlag, 1998.

[41] A.F. Smeaton, Information retrieval: still butting heads with natural language processing? in: M.T. Pazienza (Ed.), Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, Springer-Verlag Lecture Notes in Computer Science #1299, 1997, pp. 115–138.

[42] T. Strzalkowski, J. Perez-Carballo, M. Marinescu, Natural Language Information Retrieval in Digital Libraries, Proceedings of the 1st ACM International Conference on Digital Libraries (DL '96), ACM Press, 1996, pp. 117–125.

[43] T. Strzalkowski, F. Lin, Natural Language Information Retrieval, in NIST Special Publication 500-240: The Sixth Text Retrieval Conference (TREC-6), United States Department of Commerce, National Institute of Standards and Technology, 1997.

[44] R.C. Swan, J. Allan, Aspect Windows, 3-D Visualizations, and Indirect Comparisons of Information Retrieval Systems, Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1998.

[45] A. Tombros, M. Sanderson, Advantages of Query Biased Summaries in Information Retrieval, Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1998.

[46] P. Turney, Learning to Extract Keyphrases from Text, National Research Council Technical Report ERB-1057, 1999.

[47] B. Vélez, R. Weiss, M.A. Sheldon, D.K. Gifford, Fast and Effective Query Refinement, Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, 1997.

[48] R. Weiss, B. Velez, M.A. Sheldon, C. Namprempre, P. Szilagyi, A. Duda, D.K. Gifford, HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering, Proceedings of the Seventh ACM Conference on Hypertext, ACM Press, 1996.

[49] I.H. Witten, C. Nevill-Manning, R. McNab, S. Cunningham, A public library based on full-text retrieval, Communications of the ACM 41 (4) (1998) .

[50] I.H. Witten, A. Moffat, T. Bell, Managing Gigabytes: Compressing and Indexing Documents and Images, Van Nostrand-Reinhold, New York, 1994.

Ian H. Witten is a professor of computer science at the University of Waikato in New Zealand. He directs the New Zealand Digital Library research project. His research interests include information retrieval, machine learning, text compression, and programming by demonstration. He received an MA in Mathematics from Calgary, Canada; and a PhD in Electrical Engineering from Essex University, England. He is a fellow of the ACM and of the Royal Society of New Zealand. He has published widely on machine learning, speech synthesis and signal processing, text compression, hypertext, and computer typography. He has written several books, the latest being *Managing Gigabytes* (1999) and *Data Mining* (forthcoming), both from Morgan Kaufman.

Eibe Frank is a PhD student of Computer Science at The University of Waikato in New Zealand. He holds a degree in Computer Science from the University of Karlsruhe in Germany. His main research interests are machine learning techniques and their applications. He has published several articles on this topic in international journals and conferences; and co-authored the book *Data Mining* (forthcoming), published by Morgan Kaufman.

Gordon W. Paynter is a graduate student in computer science at The University of Waikato in New Zealand, where he works with the New Zealand Digital Library research project. His research interests include programming by demonstration, machine learning, information retrieval and visualization, and human–computer interaction. He received a Bachelor of Computing and Mathematical Sciences from Waikato, where he is studying towards his PhD.

Craig Nevill-Manning is an assistant professor of Computer Science at Rutgers, the State University of New Jersey. His research interests include interface and indexing technology for digital libraries, as well as bioinformatics and data compression. He received his doctorate from the University of Waikato in New Zealand in 1996, and spent 2 years as a post-doctoral fellow at Stanford University.

Carl Gutwin is an assistant professor of Computer Science at the University of Saskatchewan. His research interests include user interfaces for information retrieval systems, text mining techniques, information visualization, and the design and evaluation of real-time distributed groupware systems. He received BSc and MSc degrees from the University of Saskatchewan, and the PhD degree from the University of Calgary in 1997.

Semantic Search on Internet Tabular Information Extraction for Answering Queries

H. L. Wang
Department of Computer Science,
National Tsing Hua University.

wyvern@cs.nthu.edu.tw

S. H. Wu
Institute of Information Science,
Academia Sinica.

shwu@iis.sinica.edu.tw

K. K. Wang
Institute of Information Science,
Academia Sinica.

kiki@iis.sinica.edu.tw

C. L. Sung
Department of Computer Science,
National Tsing Hua University.

clsung@rtlab.cs.nthu.edu.tw

W. L. Hsu
Institute of Information Science,
Academia Sinica.

hsu@iis.sinica.edu.tw

W. K. Shih
Department of Computer Science,
National Tsing Hua University.

wshih@cs.nthu.edu.tw

ABSTRACT
Although extracting information from tables is essential for Internet information agents, most tables are designed for human eyes and their layout and semantic meanings are not well defined. In practice, encoding the layout of each information source is impossible. This work presents a novel semantic search approach capable of extracting information from general tables. Semantic ontology allows our agents to read tables in the same knowledge domain with different layouts. In addition, a system of layout syntax and a set of transformation rules are defined to transform tables into databases without losing their semantic meanings.

Keywords
Information Extraction, Data Mining, Semantic Query, and Table Understanding.

1. INTRODUCTION
Much useful information is presented in tabular form on the Internet. Extracting information from these tabular information sources is essential for developing Internet information agents. For example, many web sites use tabular forms to list prices. Developing a price-comparing agent requires knowledge on how to extract the price information from these web sites. The agent may need to answer questions such as "Can you tell me the web site that sells the Palm Pilot at the lowest retail price?" Conventional information retrieval technologies cannot answer this question.

A software agent can answer this question through the following steps:

Step 1. Search the web pages related to the user's queries,

Step 2. Identify all related tables in all retrieved web pages,

Step 3. Extract all necessary information from the related tables,

Step 4. Integrate information read in Step 3, and

Step 5. Answer questions in appropriate forms.

The following three sub-steps can implement step 3:

Step 3.1 Identify the semantic relation of table cells,

Step 3.2 Convert the table into data with database form, and

Step 3.3 Extract target information from data in database form by query languages.

Time	Morning	Afternoon
Mon	John Wang (2002)	Indy Lai (2005)
Tue	Jimmy Lin (2007)	Wendy Lee (2001)
Wen	Indy Lai (2005)	John Wang (2002)
Thr	Jimmy Lin (2007)	John Wang (2002)
Fri	Wendy Lee (2001)	Indy Lai (2005)
Sat	John Wang (2002)	

(a)

Time	Mon	Morning	John Wang (2002)
Time	Mon	Afternoon	Indy Lai (2005)
Time	Tue	Morning	Jimmy Lin (2007)
Time	Tue	Afternoon	Wendy Lee (2001)
Time	Wen	Morning	Indy Lai (2005)
Time	Wen	Afternoon	John Wang (2002)
Time	Thr	Morning	Jimmy Lin (2007)

(b)

Figure 1. (a) A hospital timetable. (b) The first seven reading paths of the timetable.

This work focuses on the conversion issue of tables. Our previous work concentrated on understanding the arrangement of table cells [8]. The query language in step 3.3 could be a natural language. Herein, a FAQ structure [6] is used to extract the constraints in queries and select targets from the database generated in Step 3.2.

The data mining approach in [7] must apply a learning process. In this study, the learning process is replaced by constructing a knowledge base in which all necessary ontology of natural language for reading table are included. Our approach has the potential to read tables in a website. Herein, a system of layout syntax is defined to denote the table layouts. This approach markedly differs from the approach of enumerating all possible layout features as applied by pertinent literature [1][2][3][4]. The layout syntax allows us to define a set of semantics preserving transformation rules to normalize tables with different layouts into standard database tables. For example, the table in Fig. 1 (a) is a timetable taken from a hospital's web page. Although humans can read the timetable, computers cannot. Figure 1(b) shows the corresponding reading path of each information unit, which is in a database format and comprehensible for a software agent. Although the timetable contains twelve reading paths, Fig. 1 (b) lists only the first seven.

Herein, the ability to understand a table is defined as the ability to answer queries asked by a normal user. For a table containing certain information that humans can read, the program should be able to read it as well. If humans can answer certain questions by reading a table, the program should be able to answer the questions by reading the table. However, tables can be simple or complex. The necessary semantic knowledge that an agent needs to understand a complex table may be unavailable in the ontology base. However, with some common knowledge on the common table, the agents can still extract certain information from the tables. Hence, in this paper, we propose a novel leveled knowledge schema to specify the relation between the knowledge and the information that can be extracted.

2. PROBLEM DEFINITION

This paper focuses on the following goal: "Given a table A and the related domain knowledge[1], convert table A into a database table B, while keeping the semantic equivalence of table A and table B." The complexity of tables must be investigated to achieve this goal. While Section 2.1 categorizes tables into different classes, Section 2.2 discusses how to process different classes of tables at different knowledge levels.

2.1 Table Classification

Previous works on table classification focused on extracting the table's logical structure [1][3][4]. Since this is insufficient for our purposes, this investigation classifies tables based on the semantic and layout structure knowledge.

From a semantic aspect, a **knowledge base** is used herein to identify the semantics of natural language queries and the content of tabular cells. Our knowledge base contains the ontology of **concepts** and **instances**. Each cell may have one information item. In our knowledge base, the item can be identified as either a concept or an instance of some concepts.

In the layout aspect, we propose three table classes: 1-dimensional tables, 2-dimensional tables, and complex tables. The first two classes are used more often and are simpler than complex tables. Herein, table cells are divided into labels and entries according to the identification in the semantic aspect.

1-dimensional tables: 1-dimensional tables may have one or several rows of labels above the rows of entries. The tables in a relational database are 1-dimensional table. The entries in a different column represent instances of different concepts. The labels of each column identify the categories of concepts by which we access entries in the column. For example, Fig. 2(a) illustrates the style of a normal 1-dimensional table and Fig. 2 provides an example of a 1-dimensional table.

2-dimensional tables: 2-dimensional tables have a rectangular area of similar entries. Each entry in this rectangular area represents an instance of the same concept. One or several rows of labels are above the rectangular area. These labels are instances of

other concepts. Instances in different rows belong to different concepts. At the left side of the rectangular area could be one to several columns of labels. These labels are also instances of different concepts. Instances in the same column belong to the same concept. Instances in different columns belong to different concepts. Above the columns of entries can be their concepts or empty cells. For example, Figs. 3(a)(b) show styles of 2-dimensional tables and Figs. 3(c)(d) provide illustrative examples.

			Flight	Day	Departs	Arrives	Stops/Via
C_1		C_n	LM202	Sun	11:10	13:20	0
			LM208	Mon	20:20	21:45	0
I_1^{*c}	...	I_n^{*c}	LM208	Thu	20:45	22:10	0
			LM208	Tue	20:00	22:10	0
			LM451	Mon	10:40	12:05	0
			LM963	Tue	10:40	12:05	0

(a)   (b)

Figure 2. (a) An abstract 1-dimensional table, (b) A 1-dimensional table example of (a). Where C_X denotes a string representing concept X in a cell and I_X denotes a string representing an instance of concept X in a cell. In addition, *c denotes a repetition in a column and *r denotes a repetition in a row. Moreover, ** denotes a repetition in a rectangular area. Section 3.2 provides more details.

Time	Morning	Afternoon
Mon	John Wang (2002)	Indy Lai (2005)
Tue	Jimmy Lin (2007)	Wendy Lee (2001)
Wen	Indy Lai (2005)	John Wang (2002)
Thr	Jimmy Lin (2007)	John Wang (2002)
Fri	Wendy Lee (2001)	Indy Lai (2005)
Sat	John Wang (2002)	

(a)   (b)   (c)

	Room	Mon	Tue	Wed	Thu	Fri	Sat
Morning	1	Joe Jiang	Jean Tasi	Joe Huang	Hellon Yuo	Hellon Yuo	Collin Lee
	2		Joe Jiang	Collin Lee	Jean Tasi	Joe Jiang	
	3	Jean Tasi	Collin Lee	Jean Tasi	Joe Huang	Collin Lee	Hellon Yuo
Afternoon	1	Collin Lee	Jean Tasi	Hellon Yuo	Joe Jiang	Joe Huang	
	2		Joe Jiang			Jean Tasi	
	3		Joe Huang	Collin Lee	Hellon Yuo	Collin Lee	

(d)

Figure 3. (a) (b) are abstract 2-dimensional tables, (c) is a 2-dimensional table example of (a), and (d) is a 2-dimensional table example of (b).

Complex table: In addition to 1-dimensional tables and 2-dimensional tables, there are complex tables. Complex tables can have many features. Some of those features are described as follows:

■ **Partition label:** Special labels between the data entries can make several partitions on the data entries. Each partition shares the same labels at the top of the table. For example, Figs. 4(a) and 7 illustrate this situation. In addition to that semantic identification in tables with this feature is complicated, the concept-instance relation cannot be obtained in closing cells.

■ **Over-expanded label:** For example, in Fig. 4(b), each entry I_X spans two or three I_Y under the same label C_X. In Fig. 6, there are two labels span 11 and 4 sub-labels, respectively.

■ **Combination:** Some large tables are a combination of several similar small tables. For example, in Fig. 4(c), four small tables

---

merge into a larger one. In Fig. 5, two tables are merged into a larger one.

■ **Multiple items in a cell:** Some tables may have two or more items, can be recognized as instances of the same or different concepts, placed in the same cell. For example, in Fig. 4(d), each entry has two instances in the second column. In Fig. 5, many cells have two different items: the doctor's name and the ID number of the doctor[2]. The relation between items in the same cell is referred to as inner cell relation [4].

■ **Vertical writing:** The sequence of text is presented vertically in a cell, which commonly occurs in Chinese web pages. For example, in Fig. 4(e), "ABC" and "DEF" are written vertically in a cell. However, the real sequence in the HTML source is "ADBECF".

■ **Forward reduction:** For example, comparing Fig. 4(f) and Fig. 4(g) reveals two empty cells in Fig. 4(f), which are not meaningless.

C_{X1}	C_{Xn}			
I_P				
I^{rc}_{X1} ... I^{rc}_{Xn}				
I_P				
I^{rc}_{X1} ... I^{rc}_{Xn}				

(a)

C_X	C_Z
I_Y	I_Z
$I_X I_Y$	I_Z
I_Y	I_Z
I_Y	I_Z
$I_X I_Y$	I_Z

(b)

C_X	C_Y	C_X	C_Y
I^{rc}_X	I^{rc}_Y	I^{rc}_X	I^{rc}_Y
C_X	C_Y	C_X	C_Y
I^{rc}_X	I^{rc}_Y	I^{rc}_X	I^{rc}_Y

(c)

C_X	$C_Y C_Z$
I_X	$I_Y I_Z$
I_X	$I_Y I_Z$
I_X	$I_Y I_Z$
I_X	$I_Y I_Z$
I_X	$I_Y I_Z$

(d)

A D
B E
C F

(e)

Date	Room	Doctor
Mon	1	Jenny
	2	Penny
Tue	1	David
	1	Josh

(f)

Date	Room	Doctor
Mon	1	Jenny
	2	Penny
Tue	1	David
	1	Josh

(g)

Figure 4. (a) Partition labels. (b) Over-spanned labels. (c) Combination. (d) Multiple items in a cell. (e) Vertical writing. (f) Forward reduction. (g) Original form of (f).

Morning					
Room	Mon	Tue	Wed	Thu	Fri
1	Y.C. Chen 10201	H. T. Hur 10201	H. S. Yang 10201	Y. C. Chen 10201	H. S. Yang 10201
2	J. L. Chen 10202	C. H. Lee 10202	H. T. Hur10202	H. S. Yang 10202	H. T. Hur10202
3		T. L. Tsai 10213	T. L. Tsai 10203	H. T. Hur10203	J. L. Chen 10203
Afternoon					
Room	Mon	Tue	Wed	Thu	Fri
1	T. L. Tsai 20201	L. H. Huang 20201	J. L. Chen 20201	J. L. Chen 20201	T. L. Tsai 20201
2	C. H. Lee 20202	A. C. Lai 20202			C. H. Lee 20202

Figure 5. An illustrative example of combination and multiple items in a cell

Items & Periods		Regular	Float
Fixed Deposit	3 Monthes	4.4	4.4
	6 Monthes	4.95	4.95
	9 Monthes	5.05	5.05
	1 Year	5.15	5.15
	2 Years	5.25	5.25
	3 Years	5.25	5.25
Regular Fixed Deposit	1 Year	5.25	5.25
	2 Years	5.35	5.35
	3 Years	5.35	5.35

Figure 6. A table with over-spanned labels

Rate (%)	Regular	Float
Regular Fixed Deposit		
1 Year	5.05	5.05
2 Years	5.1	5.1
3 Years	5.1	5.1
Fixed Deposit		
3 Monthes	4.35	4.35
6 Monthes	4.6	4.6
9 Monthes	4.7	4.7
1 Year	5	5
2 Years	5.05	5.05
3 Years	5.05	5.05

Figure 7. A table with partition labels

2.2 Knowledge Levels

Developing a generic algorithm to understand all tables is extremely difficult. Although certain tables can be understood with less knowledge, others cannot. In particular, tables in different knowledge domains can have different layout features. For example, consider the train timetable in Fig. 8 in which the concept *train number* can be used to access instances of its descent concepts *origin station* (KEE-LUNG) and *destination station* (PING-TUNG). There is still no explicit clue for selecting the descent concept that the string 'KEE-LUNG' should be treated as an instance of.

Herein, the understanding of table is divided into different classes based on different levels of knowledge required to understand tables. In doing so, four different levels of knowledge will be investigated.

Level 0: No table knowledge is supported. Without any table knowledge, a trivial search can be done by some keyword searching rules. However, only trivial questions can be answered.

Level 1: Preliminary table layout knowledge is supported, i.e., treat the tables as standard 1-dimensional or 2-dimensional tables. With knowledge of this level, the position of each text string can be recognized. The ability to answer questions does not markedly improve. At this level, tabular position accessing questions can be answered.

Level 2: Knowledge to distinguish between concept and instance is supported. With knowledge of this level, agents can identify the semantics of the content in each cell and the query constraints in a natural language.

Level 3: Domain knowledge to distinguish different tables is supported. With knowledge of this level, the agent can read more complex tables.

Our idea is to denote the possible layouts by **layout syntax grammar** in different table domains and use these denotations to do template matching. The matched template is used to determine the semantic of cell content. **Semantics preserving transformation** is then applied to do the layout transformation.

---

[2] Multiple items can occasionally be represented as one item if the distribution in every cell is regular. For example, the doctor's name and ID is an example of the concept: 'doctor name & id'.

Train Number						Start Time	Arrival Time	Remark
CHU KUANG	11		TAI-TUNG NEW	To	KAO-HSIUNG	05:15	07:52	Every Day
FU HSING	101	M	KEE-LUNG	To	PING-TUNG	06:03	09:08	Every Day
TZE CHIANG	1003	M	SUNG-SHAN	To	KAO-HSIUNG	07:00	09:16	Every Day
TZE CHIANG	1005	M	SUNG-SHAN	To	KAO-HSIUNG	07:40	09:46	For FRI SAT SUN only.
TZE CHIANG	1007	M	SUNG-SHAN	To	KAO-HSIUNG	07:55	10:11	Every Day
CHU KUANG	41	M	KEE-LUNG	To	CHIA-I	08:23	11:06	Every Day
TZE CHIANG	1009	M	KEE-LUNG	To	KAO-HSIUNG	08:40	10:40	Every Day
TZE CHIANG	1011	M	SUNG-SHAN	To	KAO-HSIUNG	09:00	11:15	Every Day

Figure 8. Partial view of a Taiwan train timetable

3. METHODOLOGY

3.1 Knowledge Base

Our knowledge base, constructed with a concept-sensitive model [5], contains the ontology knowledge to identify the semantics in text strings. For example, the ontology in Fig. 9(a), which is used to read the airline schedule in Fig. 9(b), contains the attributes of *flight, departure information, arrival information*, and *aircraft types*. The departure information has sub-attributes of *city* and *date-&-time*. The arrival information also has the same sub-attributes. Each attribute is a concept, and the sub-attributes are called descent concepts. Each attribute has its synonym and instances. For example, the concept of departing information may appear as "depart" or "departing". Instances of the departing city can be the name or shortcut of any international cities.

(a)
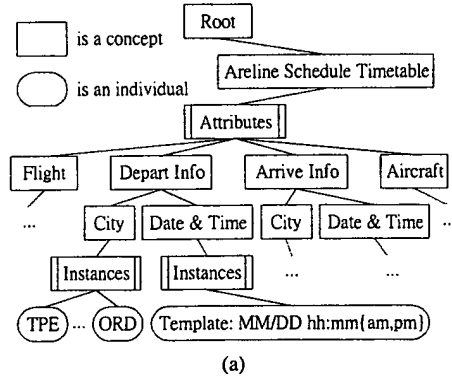
Flight	Departing		Arriving		Aircraft
	City	Date & Time	City	Date & Time	
EVA Airways 10	TPE	07/14 11:50pm	YVR	07/14 07:40pm	744
American Airlines 6647 *	YVR	07/14 09:00pm	LAX	07/14 11:45pm	737
Air Canada 9800 *	TPE	07/14 11:50pm	YVR	07/14 07:40pm	744
American Airlines 6501 *$	YVR	07/15 06:35am	LAX	07/15 09:27am	737
China Airlines 61	TPE	07/14 08:10pm	FRA	07/15 06:50am	M11
American Airlines 83$	FRA	07/15 10:40am	ORD	07/15 01:05pm	763
American Airlines 473$	ORD	07/15 02:30pm	LAX	07/15 04:35pm	738

(b)

Figure 9. (a) Ontology for airline schedule timetables, (b) an airline schedule timetable.

Ontology knowledge is first applied by template matching skills [8] to recognize possible semantic items in the text of each cell. Ambiguities might arise when the same text fragment can be

recognized as different concepts or instances of different concepts. For example, "TPE" can be recognized as instances of the departing city or the arriving city.

Fortunately, individuals often arrange items in some meaningful order and place related items close together, implying that the ambiguities can be reduced by identifying the relationships between cells. For example, in the same column, the term "departing" is recognized as departure information and the term "city" is recognized as the city of departure information and the city of arrival information. The term "city" below the term "departing", the city of departure information is more closely related to the departure information. Hence, the city of departure information for this cell is selected herein instead of the city of arrival information.

In our approach, four relationships are used to distinguish the role of each cell as a table label or table data. These four relationships are ordered by their importance and listed as follows:

(1) A concept to one of its instances, denoted as C_x-I_x;

(2) A concept to one of its descent concepts, denoted as C_x-$C_{x'}$;

(3) A concept to an instance of its descent concept, denoted as C_x-$I_{x'}$; and

(4) An instance to another instance of the same concept, denoted as I_x-I_x.

3.2 Table Layout Syntax

Our table layout syntax consists of semantic symbols and layout symbols. They are used as area notations and concatenating operators respectively.

Semantic cell notations:

C_X denotes a string represents concept X in a cell.

I_X denotes a string represents an instance of concept X in a cell.

? is a don't-care symbol, and can be any kind of cell.

Area notations:

^{*c} denotes a repetition in a column. For example, I^{*c}_X denotes Fig. 10(a).

^{*r} denotes a repetition in a row. For example, I^{*r}_X denotes Fig. 10(b).

** denotes a repetition in a rectangle area. For example, I^{**}_X represents Fig. 10(c). The first * is size of rows and the second * is size of columns.

The above * can also be replaced by numbers, variables or expressions to denote a deterministic size of repetition. For example, I^{3r}_X represent three concatenate cells of I_X in the same row.

Concatenating operator:

I is an operator capable of concatenating two sides in top-bottom direction. For example, $C_X I I^{*c}_X$ represents Fig. 10(d).

- is an operator capable of concatenating two sides in left-right direction. For example, C_X-I^{*r}_X represents Fig. 10(f).

( ) is a compound operator that guarantees the inner expression is calculated first.

246

Π represents a repetition of left-right concatenation. For example, in Fig. 10(e), $\Pi^n_{i=1} (C_{Xi}II^{*c}_{Xi}) = (C_{X1}II^{*c}_{X1})\text{-}...\text{-}(C_{Xn}II^{*c}_{Xn})$ and $I^{nr}_X = \Pi^n_{i=1}I_{Xi}$.

Σ represents a repetition of top-bottom concatenation. For example, in Fig. 10(g), $\Sigma^n_{i=1}(C_{Xi}\text{-}I^{*r}_{Xi}) = (C_{X1}\text{-}I^{*c}_{X1})\text{I}...\text{I}(C_{Xn}\text{-}I^{*c}_{Xn})$ and $I^{nc}_X = \Sigma^n_{i=1}I_{Xi}$.

Definition: A monotonic area is a rectangular area of cells that have the same concept, implying that they are strings representing the same concept or they are strings representing instances of the same concept. Monotonic areas are C_X, I_X, C^{*r}_X, I^{*r}_X, C^{*c}_X, I^{*c}_X, C^{**}_X, and I^{**}_X.

Lemma: If a table can be recursively binary partitioned until each partition is a monotonic area, then this table can be represented by table layout syntax.

Proof: For each partition P, a binary partition produces two partitions P_X and P_Y. Assume that P_X and P_Y can be represented by the layout syntax $G(P_X)$ and $G(P_Y)$, respectively. Since P_X and P_Y are produced by a binary partition, they are either concatenated from top to bottom or from left to right. Hence, $G(P) = P_X|P_Y$ if they are concatenated from top to bottom. $G(P) = P_X\text{-}P_Y$ if they are concatenated from left to right.
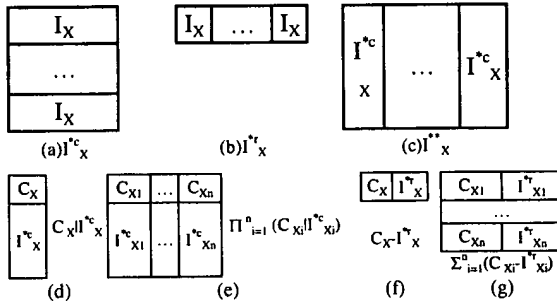

Figure 10. Areas represented by layout syntax grammar.

3.3 Semantics Preserving Transformation

The semantics preserving transformation is a mapping from a layout to another layout, while keeping the semantic relations between cells. This transformation aims to transform the arrangement from 1-dimensional or 2-dimensional tables into a standard database. A standard database can assist the storage of data and compare data from different information resources.

Herein, Γ is defined to be a set of layouts. For example, $\underset{i=1,3}{\Gamma} (I_{Xi}\text{-}$

$I^{*r}_Y)$ is a set of tables: $I_{X1}\text{-}I^{*r}_Y$, $I_{X2}\text{-}I^{*r}_Y$, $I_{X3}\text{-}I^{*r}_Y$. For 1-dimensional and 2-dimensional tables, the following two transformation rules can be used to derive the corresponding database table.

Table 1. Table transformation rules

1-Dimensional Table	From	$C_X\text{-}I^{*r}_X$
	To	$C_X II^{*r}_X$
2-Dimensional Table	From	$(?II^{(m-h)c}_X)\text{-}(?^{(h-1)*}II^{*r}_YII^{(m-h)*}_Z)$
	To	$\underset{i=1,(m-h)}{\Gamma} (?^{(h-1)*}II_{Xi}II^{*r}_YII^{(m-h)*}_Z)$

3.4 Natural Language Query

The ability to concisely recognize the query constraints, output fields, and target tables in the natural language (*NL*) query input would allow us to implement this *NL* query by a SQL SELECT command "SELECT <output fields> from <target table> where <constraints>". Three major questions must be answered: (a) How can we recognize the necessary information in a *NL* query? (b) How can we obtain a systematic approach to define tables and its fields in order to extract information and answer queries? and (c) How can we manage all database tables created to store the information extracted from web tables?

First, the field names of a database table must be defined. The fields of query constraints and the fields of the target database table must have the same name if they represent the same meaning. For example, for queries on the airline schedule timetables, the "when ... depart?" in a query and the "departure time" in a table access the same concept. In the proposed approach, the concepts are unique identities with respect to their various representations. In addition, the ontology in our knowledge base can be applied to recognize concepts and their instances in text strings, and make no difference on queries and table cells. For implementation, directly applying the concept identity string in our knowledge base as the database field names is an effective approach. Hence, field names are concept identity strings, and field values are their instance values. The query constraints are instance values and output fields are concepts. In this manner, problem (b) can be solved.

Problem (a) can be solved by FAQ structures in [6]. In the FAQ structure, keywords, concepts, instances, and their order are denoted for pattern matching. The full matched FAQ structure is selected to recognize the necessary information in the input query. The FAQ structure also contains a schema to control the set of query constraints, output fields, and the target table. The target table and query constraints are set from matched instances. In an intelligent agent, users might occasionally ask questions interactively. Hence, some information from previous interactions must be known.

Problem (c) is an information integration problem. All of the extracted information cannot be simply merged in the same database table for the same ontology because some extra information may be required, such as constraints; however, they do not appear in the input tables. For example, their timetables do not contain the ownership information of the hospital name and address. Simply merging the information extracted from timetables of all hospitals and ignoring their affiliation would not allow us to distinguish the ownerships from the query results. Hence, information extracted by the same ontology might be stored in individual tables. In addition, extra information may be required to select the target tables before accessing the data in these tables. Importantly, the information extracted outside the tables must be integrated using information integration approaches [9].

4. EXAMPLE[4] AND EXPERIMENT

Consider the table in Fig. 11(a). Its class can be recognized and, then, it can be transformed into a standard database. Applying the

---

[3] All the nodes in our knowledge bases have their unique identities.

[4] All the examples in the paper are in Chinese. We translate them in to English for international readers' convenience.

ontology in Fig. 12 allows us to recognize the semantics of cells in this table as shown in the table of Fig. 11(b). In this table, cells are grouped into four monotonic areas as shown in the table of Fig. 11(c). Applying the transformation mechanism in section 3.3 for 2-dimensional tables, we rearrange the cells in the form of Fig. 11(d). Finally the table in Fig. 11(a) is transformed into the database format table in Fig. 11(e). In the database format table, every row is a reading path of the table in Fig. 3.

The effectiveness of the proposed approach is demonstrated by selecting tables from twenty three web pages. In our experiment, the above two semantics preserving transitions in section 3.3 are used to transform the database format. Herein, only one ontology is used to identify the semantic information for each cell. Among our twenty three input tables, thirteen are 2-dimensional tables, and ten are complex ones. Figure 13(a) summarizes our preliminary results, which are highly promising for 2-dimensional tables. In addition, a success ratio of 84.62% is achieved for 2-dimensional tables. For 2-dimensional tables, eleven of them can be transformed into database tables without any error. Another one failed due to too many empty cells and the other one failed due to the lack of ontology knowledge to recognize necessary information for some cells. However, all of the ten complex tables failed. They contain features of combination, partition, vertical writing, forward reduction, and errors. The features of combination, partition, and vertical writing can be removed by a preprocessing. To demonstrate that our approach can be extended to complex tables, we apply a hand preprocessing to remove these features on eight related complex tables. Among these tables, seven can be transformed into database format correctly. The total success ratio increased to 70%. The total success ratio increased from 47.83 to 78.26. The other features require advanced knowledge level for further recognition.

Without preprocessing			
Cases	Success	Fail	Rate (%)
2D	11	2	84.62
Complex	0	10	0.00
Total	11	12	47.83

(a)

With preprocessing			
Cases	Success	Fail	Rate (%)
2D	11	2	84.62
Complex	7	3	70.00
Total	18	5	78.26

(b)

Figure 13. (a) Experimental results without preprocessing, (b) experimental results with preprocessing to remove features for complex tables.

5. CONCLUSION

This work demonstrates how intelligent agents can extract the tabular information for answering queries. With the assistance of ontology knowledge, the intelligent agents can distinguish concepts and instances in each table cell. With the layout syntax grammar, intelligent agents can recognize a particular layout. In addition, the semantics preserving transformation is developed to transfer a table into the standard database form. This paper also presents a picture of investigation on the tables with respect to different table classes crossing different knowledge levels.

6. REFERENCES

[1] Xinxin Wang (1996), "Tabular abstraction, editing, and formatting," PhD Thesis, Department of Computer Science, University of Waterloo.

[2] Hurst, Matthew & Douglas, S. (1997). "Layout and Language: Preliminary investigations in recognizing the structure of tables," Proceedings of the Fourth International Conference on Document Analysis and Recognition, 28-31 August, Ulm, Germany.

[3] Douglas, Shona, Hurst, M., & Quinn, D. (1995). "Using Natural Language Processing for Identifying and Interpreting Tables in Plain Text," Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval, pp.535-546, Las Vegas, Nevada. University of Nevada.

[4] Hurst, Matthew (1999), "Layout and Language: Beyond Simple Text for Information Interaction - Modelling The Table," Proceedings of The 2nd International Conference on Multimodal Interfaces, Hong Kong

[5] Wen-Lian Hsu, Yi-Shiou Chen and Yuan-Kai Wang (1998), "A Context sensitive model for concept understanding," Proceedings of ITALLC 98, pp.161-169.

[6] Wen-Lian Hsu, Yi-Shiou Chen and Yuan-Kai Wang (1999), "Natural language agents - An agent society on the Internet," Proceedings of PRIMA 99.

[7] Chun-Nan Hsu and Chien-Chi Chang. "Finite-State Transducers for Semi-Structured Text Mining (1999)," Proceedings of IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications, Stockholm, Sweden.

[8] Huei-Long Wang, W. L. Hsu, Y. S. Chen, T. L. Lau, C. H. Tang, H. M. Yeh, W. K. Shih (1999), "A Streamlined Approach for Tabular Information Extraction," Proceedings of NCS99.

[9] Chun-Nan Hsu and Craig A. Knoblock. "Semantic Query Optimization for Query Plans of Heterogeneous Multi-Database Systems," IEEE Transactions on Knowledge and Data Engineering, 1999.
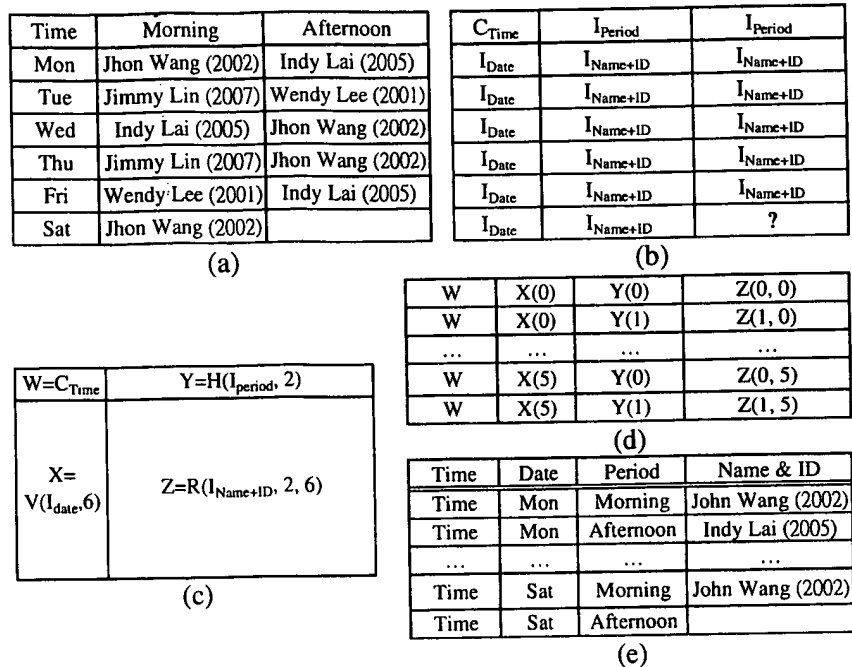
Time	Morning	Afternoon
Mon	Jhon Wang (2002)	Indy Lai (2005)
Tue	Jimmy Lin (2007)	Wendy Lee (2001)
Wed	Indy Lai (2005)	Jhon Wang (2002)
Thu	Jimmy Lin (2007)	Jhon Wang (2002)
Fri	Wendy Lee (2001)	Indy Lai (2005)
Sat	Jhon Wang (2002)	

(a)

C_{Time}	I_{Period}	I_{Period}
I_{Date}	$I_{Name+ID}$	$I_{Name+ID}$
I_{Date}	$I_{Name+ID}$	$I_{Name+ID}$
I_{Date}	$I_{Name+ID}$	$I_{Name+ID}$
I_{Date}	$I_{Name+ID}$	$I_{Name+ID}$
I_{Date}	$I_{Name+ID}$	$I_{Name+ID}$
I_{Date}	$I_{Name+ID}$	?

(b)

$W=C_{Time}$	$Y=H(I_{period}, 2)$
$X=V(I_{date}, 6)$	$Z=R(I_{Name+ID}, 2, 6)$

(c)

W	X(0)	Y(0)	Z(0, 0)
W	X(0)	Y(1)	Z(1, 0)
...	...	...	...
W	X(5)	Y(0)	Z(0, 5)
W	X(5)	Y(1)	Z(1, 5)

(d)

Time	Date	Period	Name & ID
Time	Mon	Morning	John Wang (2002)
Time	Mon	Afternoon	Indy Lai (2005)
...	...	...	...
Time	Sat	Morning	John Wang (2002)
Time	Sat	Afternoon	

(e)

Figure 11. (a) The original table, (b) The outcome after the step 3.1, (c) 4 monotonic areas, (d)(e) Transformed from Fig. 11(a) by applying the transformation rules in section 3.3
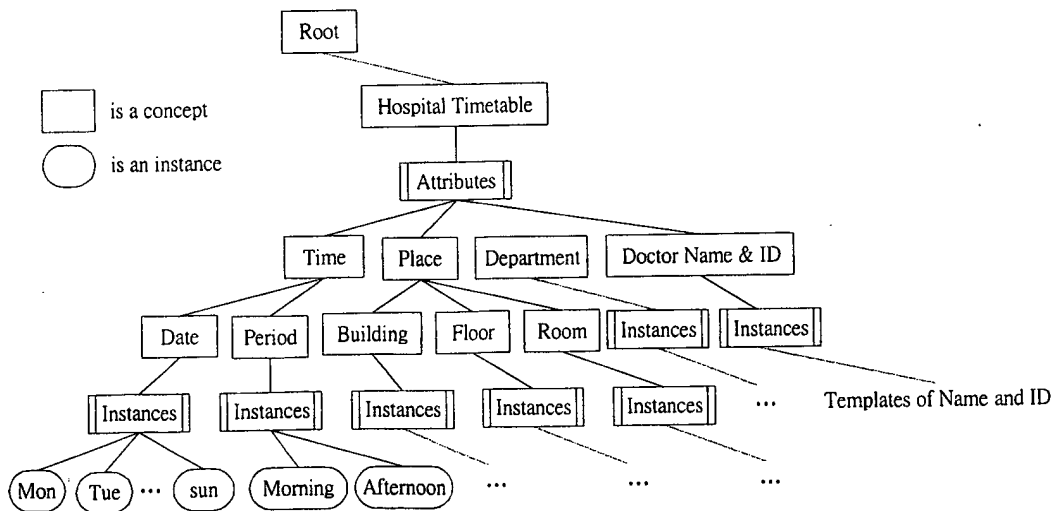

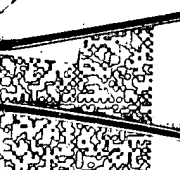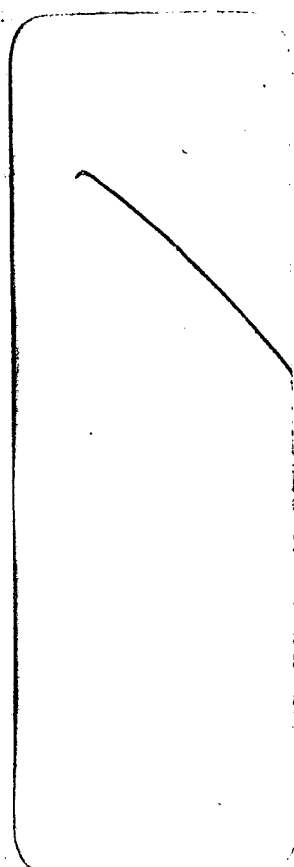**Figure 12. Ontology for the hospital timetables**

UTE

Alexandria, VA 22313-1450
If Undeliverable Return in Ten Days

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

EQUAL OPPORT

AN
AS ADDRESSED
UNABLE TO FORWARD

NOT DELIVERABLE
AS ADDRESSED
UNABLE TO FORWARD

RECEIVED

JAN 23 2008

USPTO MAIL CENTER

UNITED STATES POSTAGE
U.S. OFFICIAL MAIL

02 1A
000420447B
MAILED FROM ZIP CODE 223 4

$04.60